NAVAL POSTGRADUATE SCHOOL
Monterey, California

DTIC
ELECTE
JAN 10 1995
B

THESIS

ANALYSIS, DESIGN AND IMPLEMENTATION OF A DATABASE SYSTEM FOR THE SYSTEMS MANAGEMENT CURRICULUM OFFICE

by

Sufian I. Althawadi
Barry D. Hubbard

September 1994

Thesis Advisor: William B.Short
Co-Advisor: Shu Liao

Approved for public release; distribution is unlimited.

19950109 089

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September, 1994	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE Analysis, Design and Implementation of a Database System for the Systems Management Curriculum Office

5. FUNDING NUMBERS

6. AUTHOR(S) Sufian I. Althawadi and Barry D. Hubbard

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING RGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT *(maximum 200 words)* The Systems Management Curricular Office at the Naval Postgraduate School is burdened with the enormous administrative task of managing files for over 500 students. In a time of drastic military downsizing and funding cuts, this task will require more work of a smaller staff with less money. The burden of paper management could be lessened through automation of record keeping, while increasing efficiency and effectiveness. Valuable time for the students could be saved through elimination of excessive paperwork which they were required to prepare. Based on requirements from the Systems Management Curricular Office, this thesis designs and implements a database management system. The primary objective is to allow the incoming class of students to enroll using this system instead of traditional paper forms, enabling the staff to focus on more non-administrative tasks. This system will store, sort and compare data relevant to all students while minimizing the need to maintain hardcopy files. Additionally, the staff will be able to query reports and generate letters with minimal effort . The system is also analyzed to determine possible enhancements that could be added in the future. The Systems Management Database Systems (SMDS) is designed using Borland's PARADOX version 4.0.

14. SUBJECT TERMS Database management system (DBMS), DBMS design, DBMS development, DBMS implementation, PARADOX ver 4.0, systems management database system (SMDS)

15. NUMBER OF PAGES 142

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

Approved for public release; distribution is unlimited.

ANALYSIS, DESIGN AND IMPLEMENTATION OF A DATABASE SYSTEM FOR THE SYSTEMS MANAGEMENT CURRICULUM OFFICE

Sufian I. Althawadi
First Lieutenant , Bahrain Army
B.S., University of Bahrain, 1987

Barry D. Hubbard
Lieutenant Commander, United States Navy
B.S. U.S. Naval Academy, 1981

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 1994

Authors: _____
Sufian I. Althawadi

_____
Barry Hubbard

Approved by: _____
William B. Short, Thesis Advisor

_____
Shu Liao, Co-Advisor

_____
David R. Whipple, Chairman
Department of Systems Management

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution
Availability Codes
Dist | Avail and/or Special
A-1

ABSTRACT

The Systems Management Curricular Office at the Naval Postgraduate School is burdened with the enormous administrative task of managing files for over 500 students. In a time of drastic military downsizing and funding cuts, this task will require more work of a smaller staff with less money. The burden of paper management could be lessened through automation of record keeping, while increasing efficiency and effectiveness. Valuable time for the students could be saved through elimination of excessive paperwork which they were required to prepare.

Based on requirements from the Systems Management Curricular Office, this thesis designs and implements a database management system. The primary objective is to allow the incoming class of students to enroll using this system instead of traditional paper forms, enabling the staff to focus on more non-administrative tasks. This system will store, sort and compare data relevant to all students while minimizing the need to maintain hardcopy files. Additionally, the staff will be able to query reports and generate letters with minimal effort . The system is also analyzed to determine possible enhancements that could be added in the future. The Systems Management Database Systems (SMDS) is designed using Borland's PARADOX version 4.0.

TABLE OF CONTENTS

I. INTRODUCTION 1

A. BACKGROUND 1

B. SYSTEMS MANAGEMENT DATABASE SYSTEM (SMDS) 2

C. CHAPTER DESCRIPTIONS 2

II. SYSTEM DEVELOPMENT (GENERIC) 3

A. PHASE I: DEFINITION PHASE 3

1. Form Team 3

2. Define Problem 3

3. Establish Scope 3

4. Assess Feasibility 4

B. PHASE II: REQUIREMENTS PHASE 4

1. Create Data Model 4

2. Determine Update, Display and Control Mechanisms 4

3. Interview Users 4

4. Use Prototypes 5

C. PHASE III: EVALUATION PHASE 5

1. Select Systems Architecture 5

2. Reassess Feasibility 5

3. Reassess Requirements 5

D. PHASE IV: DESIGN PHASE 6

1. Develop Database Design 6

2. Develop Application Design 6

E. PHASE V: IMPLEMENTATION 7

1. Construct Database 7

2. Build Application 7

3. Testing 7
4. Installation 7
5. Maintenance 8

III. SYSTEM DEVELOPMENT (SMDS) 9
A. PHASE I: DEFINITION PHASE 9
B. PHASE II: REQUIREMENTS PHASE 11
C. PHASE III: EVALUATION PHASE 15
D. PHASE IV: DESIGN PHASE 16
E. PHASE V: IMPLEMENTATION 18

IV. CONCLUSIONS 21

REFERENCES 23

APPENDIX A. Entity Relation Diagram 25
APPENDIX B. Data Dictionary 27
APPENDIX C. Data Flow Diagrams 49
APPENDIX D. Update, Display, and Control Mechanism 57
APPENDIX E. Relational Diagram 67
APPENDIX F. Menus, Forms, and Reports 69
APPENDIX G. Logic for Menus and Submenus 97

INITIAL DISTRIBUTION LIST 131

I. INTRODUCTION

A. BACKGROUND

The Naval Postgraduate School (NPS) was established to serve the educational needs of the Navy. Its specific mission is to provide advanced professional studies at the graduate level for military officers from all services as well as other nations. The NPS is fully accredited and confers master's, engineer's and doctor's degrees.

The NPS is divided into eleven academic departments and four interdisciplinary academic groups. The Systems Management (SM) curriculum is made up of one Educational Technician, one secretary, an assistant Curricular Officer, a Curricular Officer and 562 student Officers. The administrative burden of manually tracking information on all these students is inefficient and requires excessive manpower. The focus of the staff has become full time management of administrative tasks related to students. Additionally, the number of students supervised has grown because of the Information Technology Management curriculum being absorbed into Systems Management curriculum during the summer quarter of 1994.

Students are required to fill out numerous forms when first reporting to the NPS. These forms range from personal locator cards to next of kin notification cards, class schedules, course matrices and list of dependents. There are numerous redundancies. Difficulties have been noted by the SM curricular office staff in locating specific student forms and the information on the forms has often been incomplete or inaccurate.

The Curricular Officer, Systems Management department, requested a feasibility study with regards to automating the maintenance of student information and building a supporting database. It was desired that students would log on to a single terminal to input all their required data. This would enable the staff to track, sort and check data from a single source. The system would be required to generate a limited number of reports, letters, and lists while storing information for historical reference. This thesis proposes a system designed to accomplish these tasks.

B. SYSTEMS MANAGEMENT DATABASE SYSTEM (SMDS)

The Systems Management Database System (SMDS) was designed to ease the administrative burden of the Curricular Office staff while making storage and retrieval of vital information easier and more efficient. Additionally, the ability to sort like information for reports and similar tasks further streamline the tasks performed. To accomplish this we interviewed the office staff to determine their requirements.

Borlands PARADOX for Windows, version 4.0, was used to build this system. Many iterations and revisions were accomplished through recommendations and updates from the Systems Management office staff. SMDS is menu driven, designed to mimic DOD forms in as logical a fashion as possible, and designed to be user friendly for those without background in using PARADOX.

C. CHAPTER DESCRIPTIONS

Chapter II is a discussion of the generic System Development methodology considered in developing this automated information system.

Chapter III will discuss the SMDS System Development process and the phases discussed above.

Chapter IV will discuss conclusions, discussions and recommendations. This area will focus on improvements and possible areas for growth for the system.

Appendices A through G provide support and substantiation of requirements, a data dictionary, ObjectPAL text and data flow diagrams.

II. SYSTEM DEVELOPMENT (GENERIC)

The SMDS was developed using the five standard phases of the Systems Development Life Cycle (SDLC). The five phases are the definition phase, requirements phase, evaluation phase, design phase, and implementation phase. This chapter will discuss the generic requirements of each phase.

A. PHASE I: DEFINITION PHASE

1. Form Team

Simply stated, the definition phase determines what a system is to do. The initial action is to form a team of individuals to build the system. Attention should be paid to team members' strengths and levels of experience. The team should be large enough to accomplish the tasks at hand, yet not so large as to unduly influence the development process.

2. Define Problem

After the team has been formed, the problem to be solved must be defined. A problem is a perceived difference between what is and what it ought to be. Since problems are perceptions, individual definitions of the problem may vary greatly. The team must reach some agreement as to a definition and establish how far to go with a solution.

3. Establish Scope

Establishing the scope of the problem is defining the limitations of how the team can help to solve a specific portion of the defined problem. The users may want too many features or possibly not enough. The task of defining the scope establishes proposed parameters for both developers and users.

4. Assess Feasibility

After the team has been formed, the problem defined and the scope established it is necessary to determine the overall feasibility of the project. Areas to consider are cost, time, and schedule requirements.

At the end of the definition phase the team should report back to the client for feedback. Improvements or refinements can be made at this time.

B. PHASE II: REQUIREMENTS PHASE

1. Create Data Model

A requirements phase is necessary to build on the specifics laid out in the definition phase. The expansion of the definition phase is done through use of users requirements and data models. The users data model describes the objects that are to be stored in the database and denotes their relationships to one another and their structure. The requirements data model represents the basis for database design. This should be a "big picture" of input documents, processes required, and general output desired by the user.

2. Determine Update, Display and Control Mechanisms

Additionally, within the requirements phase, it is necessary to establish functional components or mechanisms to update, display, and control the database. This will define the means by which the user will maintain a current database and retrieve useful information from it.

3. Interview Users

The ultimate authority on application requirements are always the users. The development team will use its experience, background, and knowledge to help users form their requests regarding inputs, outputs, and constraints into plausible needs.

4. Use Prototypes

Mock-ups of forms, reports, and an input menu can be developed to help users envision the future product. The purpose of these prototypes is to open an avenue for dialogue between the team and the users. With appropriate feedback the team may be able to extract additional requirements from the users and further refine the system in its early stage.

The result of this phase could be a data-flow diagram, entity-relationship diagram, object diagram, various prototypes, summary of update, display, and control mechanism or any combination of these.

C. PHASE III: EVALUATION PHASE

1. Select Systems Architecture

The evaluation phase begins after all the data collected in the requirements phase is compiled and considered. During this phase a systems architecture should be selected and alternatives should be considered to ensure the ideal match is made for the user. The system initially selected may be excluded due to new information exposed in the requirements phase.

2. Reassess Feasibility

After deciding the specifics of the hardware to be used, a reassessment of its feasibility should occur. This reassessment should be more specific than that considered in the definition phase. During the reassessment considerations should include expenses, overall scope, and timing as well as any new requirements.

3. Reassess Requirements

If it appears any of the evaluated areas cannot be achieved by the development team, the users need to be notified and an effective feedback loop should ensure the project becomes achievable. It may be as simple as an adjustment to schedules, tweaking

the budget or a more major reduction in physical requirements. Another consideration may be the possible deferral or exclusion of actions.

D. PHASE IV: DESIGN PHASE

1. Develop Database Design

Application and database design will take place within the design phase. Here, the task is to meet the users' specific needs through designed programs and procedures; specifications for hardware are also written during this phase. Files are established (relation tables), data items (attributes) are defined, and relationships are correlated between objects. Relationships between objects can be simple one-to-one, one-to-many or more complex many-to-many. Normalization should be conducted to ensure there are no anomalies between relations. Elimination of anomalies occurs by splitting the relation into two or more separate relations, each containing a single theme. Objects may be a basic, simple object or a grouping of objects called an aggregation.

2. Develop Application Design

Within the design phase, the database and applications are created. An application is a collection of menus, forms, reports, and queries that enable users to interact with and update the system. Mechanisms by which the system is to be implemented and updated will be developed and the program's logic will be decided. This is the ideal time to detect errors prior to building the system. Beyond this point finding errors will be difficult and correcting them expensive.

The output of this design phase should be a relation diagram, relation definitions, menu hierarchy, and pseudo code for each menu and sub-menu.

E. PHASE V: IMPLEMENTATION

1. Construct Database

The final phase is implementation. The task at hand is to build the system according to the specifications decided to this point. Users' needs must be isolated at this juncture. Any further requirements will adversely affect the systems development. Programming usually occurs at this point. Using the data definition subsystem of the engineered DBMS, the design is converted to fit the user's requirements. The goal is to construct the system while strictly adhering to the design. Hardware is installed, programs are developed, procedures are documented, and office staff and users are trained.

2. Build Application

Forms, reports, and menus need to be built through application development, as well as construction of transaction processing programs.

3. Testing

An often ignored area of implementation is testing. Testing verifies that any errors which may have been created in the modeling or implementation phases are discovered, and that the system performs those functions as desired by the user. This testing can be accomplished in a number of ways. The testing should not be isolated to a specific phase; rather it should be distributed throughout the entire project as it progresses. The types of testing vary greatly depending on the complexity of the system and its developers.

4. Installation

Installation is one of the final steps in implementation. Installation can occur in either of four strategies. The first of these is the parallel strategy, whereby both the old and new systems operate side by side until it is proven that the new system is working

properly. The second is the pilot strategy, where only a small piece of the function or office is converted to the new system. The new system operates in one area with the old system remaining in place until conversion occurs later. Phase-in is the third strategy. Here, the old system is gradually replaced by the new system. The final strategy is direct cutover. Conversion takes place in one fell swoop, with the new system replacing the old all at once. [James A. Senn, 1990, Information Systems in Management].

User and operator guides and documentation are generated as well in this phase. Training is recommended to ensure a smooth transition from the old system to the new one. The training should be complete such that users and system administrators are familiar with what the system can and will do for them.

5. Maintenance

Maintenance requires the verification of three areas:

a. Correction of errors discovered during system operation.

b. Implementation of modifications to the system due to user requests or changes in requirements after implementation.

c. The implementation of performance enhancements and improvements to user interfaces.

It is important to maintain the system with minimal disruption to the users; therefor, a "high degree of data independence" is desired so as to insulate applications from the physical organization of the database .

III. SYSTEM DEVELOPMENT (SMDS)

A. PHASE I: DEFINITION PHASE

The development team was comprised of two military officers; Sufian Althawadi, an Army Lieutenant from Bahrain and Barry Hubbard, a Navy Lieutenant Commander in the U.S. Navy. Both were students in the Naval Postgraduate School's Information Technology Management curriculum.

A problem of locating information on students at the Naval Postgraduate School was noted by the Systems Management Curriculum Officer, during the 1994 school year. Additionally, the hours expended by the curricular office staff tracking volumes of paperwork relating to students was excessive as was storing and cataloging the paper. With the trend of downsizing in today's Navy, the desire to be able to accomplish equivalent tasks with less personnel, heightened the interest in this area.

As a consequence of the above, the SM curricular officer asked that a feasibility study be conducted to design a Database Management System (DBMS) that could be updated by individual students, maintained by one office/staff member, be available to all students, possess sufficient security so as to observe a student's privacy of information, and be installed on a single IBM compatible 386 or 486 P.C..

The scope of the issue was to build a DBMS that could replace numerous forms, cards and records required at initial student check-in. If all these bits of information could be stored in conjunction with information regarding classes, grades, schedules, and the Military's physical readiness test (PRT) then the task of managing this information could be much more efficient. Reports would be ready made, letters would automatically be generated, and greater attention could be paid to students regarding their day-to-day issues.

The goal was to build a system that would utilize the following files:

- STUDENT
- MILITARY
- EDUCATION
- SPOUSE
- CHILDREN
- FACULTY
- COURSES
- SCHEDULE
- CURRICULUM
- THESIS
- DEPARTMENT
- ADD/DROP
- PHYSICAL

Money was limited due to Navy-wide constraints on funding. In as much as money was tight a dedicated P.C. could not be made available. Instead of a stand-alone computer, the SMDS will share a P.C. with the newly installed voice mail system. The total cost from inception to implementation/testing is $5000.00 with annual maintenance expenses forecast not to exceed $1500.00. If the Navy could purchase a unique P.C. for this system it would add an additional $5000.00 to the estimate above. It was determined that, for the cost of an upgrade to the 486/DX 66MHz IBM compatible P.C.'s with 8MB memory and time related to developing this DBMS, it could be possible to design the system within the time specified.

Benefits to be gained from SMDS include:

1. Savings of numerous man-hours from automation.

2. Time savings for curriculum staff to perform other functions by speedier retrieval of information.

3. Paper reduction, resulting in cost savings as well as decreasing storage space required.

4. Quality of data entries can be reviewed more easily resulting in higher integrity of data.

5. Increased ability to sort, calculate, and conduct statistical analysis of data stored.

B. PHASE II: REQUIREMENTS PHASE

A decision of a database development style was necessary to begin. The choices were Top-Down, Bottom-Up, or a Hybrid approach which uses techniques from each style. Top-Down development was selected because entities in the Entity-relationship Diagram (E-R) were developed with a particular organizational structure in mind.

The overall goal of those interviewed was to capture the information necessary to eliminate excessive paperwork retention by the staff. Additionally, quicker, more efficient data retrieval, and sorting of information regarding each student was desired. User interviews were conducted with each member of the Systems Management (SM) curricular Office staff that will be involved in the use of the system. Those interviewed were the Educational Technician (Ed. Tech.); the Assistant Curricular Officer and the Curricular Officer. Questions posed to the SM curricular office staff were geared to solicit improvements to their quality of life in the work place while improving overall efficiency. Student information was often out of reach or difficult to retrieve for the staff. If a student's file was not where it should be, there were no back up files available. Most student information could not be compared unless it was part of the NPS FOCUS system. Even as a part of the FOCUS system, letters of caution or reprimand could not be tied together, nor automatically generated. Additionally, the PRT results and grades had to be calculated using tables, height/weight charts, and conversion factors.

The staff specifically requested the following reports:

- Student Graduate Report
- Summary of Students by:
  - Country
  - Service
- Notification of Academic Performance
- Notification of Academic Improvement
- Notification of Academic Probation
- Continuation of Academic Probation

The curricular staff additionally desired a system in which students would carry their own diskette that would represent their individual file. The student would be able to go to any P.C. with PARADOX for Windows installed and enter their own data. These diskettes would then be turned over to the SMDS systems administrator to update the master SMDS database.

Using the experience of the design team, further user requirements were developed by; (1) determining the reports and documents most utilized and (2) examining the properties which needed to be captured when modeling those reports and forms with a focus on eliminating redundancy. The forms students must fill-in at the initial registration range from a locator card, next-of-kin information, dependent information, education history, and the list goes on. The forms are not all produced at NPS, and are, for the most part generic. Because of their non-specific nature, a great deal of redundancy exists between documents. Within the DBMS the system will help identify and hopefully eliminate this redundancy. Subtle changes to these forms were made for ease of use, elimination of redundancy as well as for overall esthetics.

Through numerous interviews with the above personnel the following twenty objects or entities were decided upon. These objects represent the most specific entities possible and are discussed below.

- STUDENT

- MILITARY
- EDUCATION
- SPOUSE
- SPOUSE ACTIVE DUTY
- CHILDREN
- FACULTY
- COURSES
- SCHEDULE
- WEEKDAY
- PERIODS
- STUDENT SCHEDULE
- CURRICULUM
- CURRICULUM COURSES
- REQUIRED COURSES
- THESIS
- PHYSICAL
- PHYSICAL CHART
- DEPARTMENT
- SUBSPECIALTY

The STUDENT entity is the central object. STUDENT will hold all the personnel information to include a scanned photo of those enrolled in the SM curriculum. This will act as a lookup table for all other tables, in other words, all information should be entered in this table first to facilitate the use of all other tables. MILITARY is a weak entity of STUDENT containing professional information tied to a student's military career, past, present, and future. A weak entity is an entity that is dependent upon another entity or a "parent" for its own identity. EDUCATION is a weak entity of STUDENT and collects information related to a student's past education. SPOUSE is a weak entity of STUDENT and contains the spouse's personal data and addresses where they reside.

SPOUSE ACTIVE DUTY is a weak entity of SPOUSE and discusses the likelihood that a student's spouse is also an active duty member of the armed forces, containing personnel data unique to their military affiliation. CHILDREN is a weak entity of STUDENT and are those dependents of the student. It records data relating to their date of birth and gender. FACULTY identifies instructors at the Naval Postgraduate School and vital data relating to their ID number, department code, office room, phone number(s), and E-mail address. COURSES is an entity describing course number, title, and credits for those classes taught at NPS, much like the NPS catalog. SCHEDULE is described by course number, segment number, faculty ID and room number in which the course is taught on a quarterly basis. WEEKDAY is a weak entity of SCHEDULE and contains the date the course is offered. PERIODS is a weak entity of WEEKDAY and lists the periods in which the class is offered. STUDENT SCHEDULE is a weak entity of STUDENT made up of student SSN, course number, segment number, quarter and course type for the current quarter only. The entity CURRICULUM denotes curriculum number, title; whether it is 3000 or 4000 level class; the number of quarters required for that curriculum; and the academic associate, subspecialty and curricular office codes associated. CURRICULUM COURSES is a weak entity of CURRICULUM and contains curriculum number and quarter ordered for a particular curriculum. REQUIRED COURSES is a weak entity of CURRICULUM COURSES and is distinguished from other courses in that they are necessary for completion of a specific degree. REQUIRED COURSES lists curriculum number, curriculum quarter and course number and type (other courses would be optional or elective in nature). THESIS is a weak entity of STUDENT that documents the thesis topic, thesis advisor and the students forwarding address to send them their masters degree. PHYSICAL is weak entity of STUDENT made up of statistics documenting a student performance on the military physical readiness test (PRT). PHYSICAL has no true relation to the pursuit of a masters degree, but it is a requirement for active duty students to pass a PRT on a twice per year basis. PHYSICAL CHART is a weak entity of PHYSICAL that stores the PRT point chart that

compares actual sit-up, push-ups, run and swim results with point totals. DEPARTMENT is an entity that is described with a department code and name and includes all academic departments. SUBSPECIALTY is an entity describing subspecialty codes correlating to individual graduate degrees.

The above entities are displayed as an E-R diagram in appendix A. The entity and domain definitions are displayed in the data dictionary in appendix B.

The collection of data flow diagrams are displayed in appendix C. These diagrams describe the overall flow of the information in the system, and the lower level processes. Attributes and functions of the SMDS are listed with summaries of update, display and control mechanism in appendix D by input, output and process notes.

The SMDS should have a back-up tape drive to ensure information integrity and needs to be backed up on a daily basis for database files, or quarterly basis to down load previous quarters information and up load the next quarters. The SMDS should also be equipped with a restore function that enables the system administrator to reestablish all data lost due to catastrophe, operator error or virus. Restoration can be done globally or for individual database files.

C. PHASE III: EVALUATION PHASE

The system selected was a 486 IBM PC compatible. This PC was selected because it was readily available to the SM staff and funding constraints prohibited any large expenditures. The system is currently located in the SM office. Because of the single PC operation it was decided the DBMS would reside on the PC and each student would have their own disk representing their individual student file. The SM Ed. tech. will act as systems administrator.

During a reassessment of requirements it was decided that the yet-to-be announced new PRT format would not be included. As a result, a height weight chart would be used in lieu of the default percent body fat calculations.

D. PHASE IV: DESIGN PHASE

Logical database design centers around the primary entity STUDENT. The entities MILITARY and PHYSICAL are weak entities of STUDENT with a one-to-one relationship. In both cases the key of the STUDENT entity (SSN) is stored in the MILITARY and PHYSICAL entities. PHYSICAL CHART is a weak entity of PHYSICAL with a one-to-many relationship. PHYSICAL CHART is used to lookup values (Curl-ups, Push-ups, Run and Swim) related to the PRT. The entities THESIS, SPOUSE, CHILDREN and EDUCATION are also weak entities of STUDENT but with a one-to-many relationship. Here, the key of STUDENT is stored in these weak entities. ACTIVE DUTY is a weak entity of SPOUSE utilizing the spouses SSN as its key, with its primary attributes being Rank, Service and home address. The SSN in all the weak entities listed above also represents a foreign key to those entities. The data in the STUDENT entity drives all other entities and must be completed before effectively utilizing the weak entities.

The CURRICULUM entity has Curr. No and P Code as its key and has a one-to-many relationship with the entity SUBSPECIALTY. SUBSPECIALTY's key is P Code with an attribute of a title. CURRICULUM COURSES is a weak entity of CURRICULUM with a one-to-many relationship with CURRICULUM, and has Curr. No (foreign key) and order as its composite key. CURRICULUM COURSES is linked with the entity COURSES by the relation REQUIRED through a many-to-many relationship. The relation REQUIRED has as its keys Order, Curr. No. and Course number. These are the keys of COURSES and CURRICULUM COURSES combined.

STUDENT entity is connected to the CURRICULUM entity through a many-to-many relationship entitled ENROLLED. The keys of STUDENT and CURRICULUM become the composite key of ENROLLED. ENROLLED attributes are Section No and graduation date.

STUDENT entity is connected to the COURSES entity through a many-to-many relationship entitled TAKENBY. The keys of STUDENT and COURSES become the composite key of TAKENBY. TAKENBY attributes are Grade and Course type.

STUDENT entity is connected to the SCHEDULE entity through a many-to-many relationship entitled ADD/DROP. The keys of STUDENT and SCHEDULE become the composite key of ADD/DROP. ADD/DROP attributes are Date and type of transaction.

STUDENT entity is connected to the SCHEDULE entity through a many-to-many relationship entitled STUDENT SCHEDULE. The keys of STUDENT and SCHEDULE become the composite key of STUDENT SCHEDULE. STUDENT SCHEDULE attributes are Quarter Order and Course type.

FACULTY entity is connected to the SCHEDULE entity through a one-to-many relationship. The keys of FACULTY (Faculty ID) is stored in SCHEDULE. FACULTY entity is connected to the DEPARTMENT entity through a one-to-many relationship. The key of DEPARTMENT (Dept. Code) is stored as an attribute in FACULTY.

The entity SCHEDULE has a one-to-many relationship with its weak entity WEEKDAY. Attributes of the entity WEEKDAY are Course No., segment and day.

The above entities and relationships are graphically represented in the Relational Diagram, Appendix E and the relation definitions, Appendix B.

Menus, forms and reports are listed in Appendix F.

Menus: Figure 1 is the login screen, requiring the last name of the user and a predetermined password. Figure 2 is the main menu listing all the selections possible for the SMDS. Figure 3 represents the student submenu and allows access to eight possible selections. Figure 4 is the curriculum submenu. Figure 5 represents the student schedule submenu. Figure 6 represents the schedule submenu. Figure 7 represents the generate reports submenu. Figure 8 represents the performance letters submenu. Figure 9 represents the codes submenu. Figure 10 represents back up submenu. Figure 11 represents restore submenu. Figure 12 represents a submenu for selecting a table to

restore. Figure 13 represents student disk submenu. Figure 14 represents a submenu for restore (student disk). Figure 15 represents a submenu of select a table to restore (student disk).

Forms: Figure 16 represents student form. Figure 17 represents military form. Figure 18 represents spouse form. Figure 19 represents children form. Figure 20 represents physical form. Figure 21 represents education form. Figure 22 represents faculty form. Figure 23 represents courses form. Figure 24 represents master schedule form. Figure 25 represents course schedule form. Figure 26 represents curriculum form. Figure 27 represents curriculum courses form. Figure 28 represents enrolled in form. Figure 29 represents student schedule form. Figure 30 represents add/drop form. Figure 31 represents academic record form. Figure 32 represents thesis form. Figure 33 represents password form. Figure 34 represents password change form. Figure 35 represents department form. Figure 36 represents subspecialty form. Figure 37 represents physical chart form.

Reports: Figure 38 represents the individual report menu. Figure 39 represents the group report menu. Figure 40 represents notification of academic performance report. Figure 41 represents notification of academic improvement report. Figure 42 represents notification of academic probation report. Figure 43 represents continuation of academic probation report. Figure 44 represents Summary of students by: country and service report. Figure 45 represents Student graduate report.

The logic (pseudo code) for the menus and submenus are coded in ObjectPal and is listed in Appendix G.

E. PHASE V: IMPLEMENTATION

Borlands PARADOX for Windows was selected to build the Systems Management Database System (SMDS). PARADOX was suited to handle the tasks at hand, as well as having sufficient power to implement additional features that would streamline tasks. Using the information discussed in the previous phases, the database

tables were constructed. The next task in creating the database data was to ensure that the data and database were compatible (referential integrity). In other words, if data entered in the database key fields were changed all the corresponding fields in the dependent tables would change accordingly.

The system hardware parameters are: an IBM PC compatible 386 (or greater) with PARADOX for Windows ver 4.0 or later installed. The SMDS requires four 3.5" disks to hold all the systems information (menus, forms, db files, and reports).

Forms, reports and menus were constructed using Pal (PARADOX's screen painter) to closely simulate those prototype forms and reports provided by the users at the outset of this project. Furthermore, ObjectPal (PARADOX's data manipulation language) was used to provide links, correlations and relationships between entities.

A pilot strategy for conversion was decided upon. This strategy should be most effective due to the user's desire to gradually implement the system as new sections of students arrive. The pilot strategy is most applicable where no previous automated system existed and a new system is to be implemented. This technique is safer than others because it minimizes the risk to the user should any problems occur in implementation since the old system is still up and running. The old system of maintaining files should gradually decrease as the SMDS becomes more utilized.

Training and familiarization sessions were held with the SM staff to enable them to become comfortable with the system's capabilities. A users guide has been published and promulgated to help individuals navigate through SMDS and is listed in appendix H. Due to the system's simplistic nature of push button screens and menus it was not necessary to conduct more extensive training for the users. The Ed Tech received additional training in system installation, troubleshooting, back-up techniques and a deeper level of overall expertise.

Testing entailed the entry of pre-assigned data selected to test the known ranges of the SMDS. The intent was to see if a student could push the system beyond its known parameters and to a point not expected by the system developers. Files were selected for

students that had academic performances ranging from outstanding to extremely poor. When the data was entered, the system was then queried to see if the results were as expected. Twenty student records were selected by the SM department staff. After entering the data into the database, normal queries to the system and generated reports were compared to those conducted manually. Modifications to the system were able to correct errors as they were detected.

Maintenance will be primarily managed by the system administrator. The plan will be for the administrator to ensure that any repetitive errors are noted and corrected through manipulation of ObjectPal or Pal as needed. Additionally, the systems administrator will be able to implement modifications to SMDS as the needs arise; again with the use of ObjectPal. As time passes, the staff and users may develop performance enhancers or additional bells and whistles; this too will fall on the system administrator's shoulders. Data independence was established at the highest degree possible to ensure the application would experience a minimum of disruption.

IV. CONCLUSIONS

The SMDS system is on-line and ready for the incoming section of students. The SMDS was successful in meeting the expectations of the users in that the system can do what it was designed to do. The SM staff will be able to learn more regarding the extensive and additional capabilities of the SMDS as time goes on. The staff will soon notice an enhancement to their work environment due to the SMDS.

It was noted that as the team tried to follow the conventional SDLC methodology the tendency was to merge requirements phase items with the design phase and visa-versa. The only way to keep these steps separate was to constantly review the SDLC outline and re-read reference material. One part of the problem is the users don't usually know the different SDLC phases; and in their discussions they lump all their needs and information into one pot. The job of the designers has to start by sorting the users information into its proper phase and proceed from there. It also is difficult to resist the urge to do parts of each phase and then go back and fill-in afterwards. This too should be avoided because of the risk of missing something in this haphazard style.

An observation of data proprietorship by the registrars office led to a suggestion for future system improvement. The data this system will rely upon will in large part come from the FOCUS database. If this information could be provided to the systems administrator in dBase, ASCII or any other PARADOX format the task of data entry would be simplified greatly. SMDS may be a good candidate for an application to be placed on the NPS network to facilitate data entry at the students level. This step would eliminate the need for the systems administrator to transfer data from student disks to the main database. If the system were to be placed on the NPS network system it would be necessary to bring the individual password/ security system on-line. It wasn't necessary in the initial phase because each student will hold his or her own disk and will not be accessible to others. Upgrading from PARADOX for Windows version 4.0 to version 5.0 will provide the SMDS with the latest innovations and greater power. This upgrade

should be available since the SM department is a registered owner of PARADOX 4.0. Floppy disks will not easily hold all the data anticipated in SMDS. A tape back-up system would ensure a state-of-the art redundancy system; therefore, the cost of purchasing a tape back-up would be worthwhile.

After a semester of evaluating the system, a review of the initial requirements should be conducted. As users note new needs and capabilities the systems ObjectPal can be manipulated to increase reports, letters, and information sorts to better suit their needs. These changes/additions could be simple enough to be accomplished by the systems administrator; but may be extensive enough to warrant a follow-on effort by another thesis student.

It is recommended that rather than having the system administrator effect a change every time one is decided upon, he maintain a log of proposed changes for implementation at a specific time. This will be a labor saving device as well as enable the curricular officer to review the changes before they are made. After all the new changes are entered a new version of the system would then exist.

REFERENCES

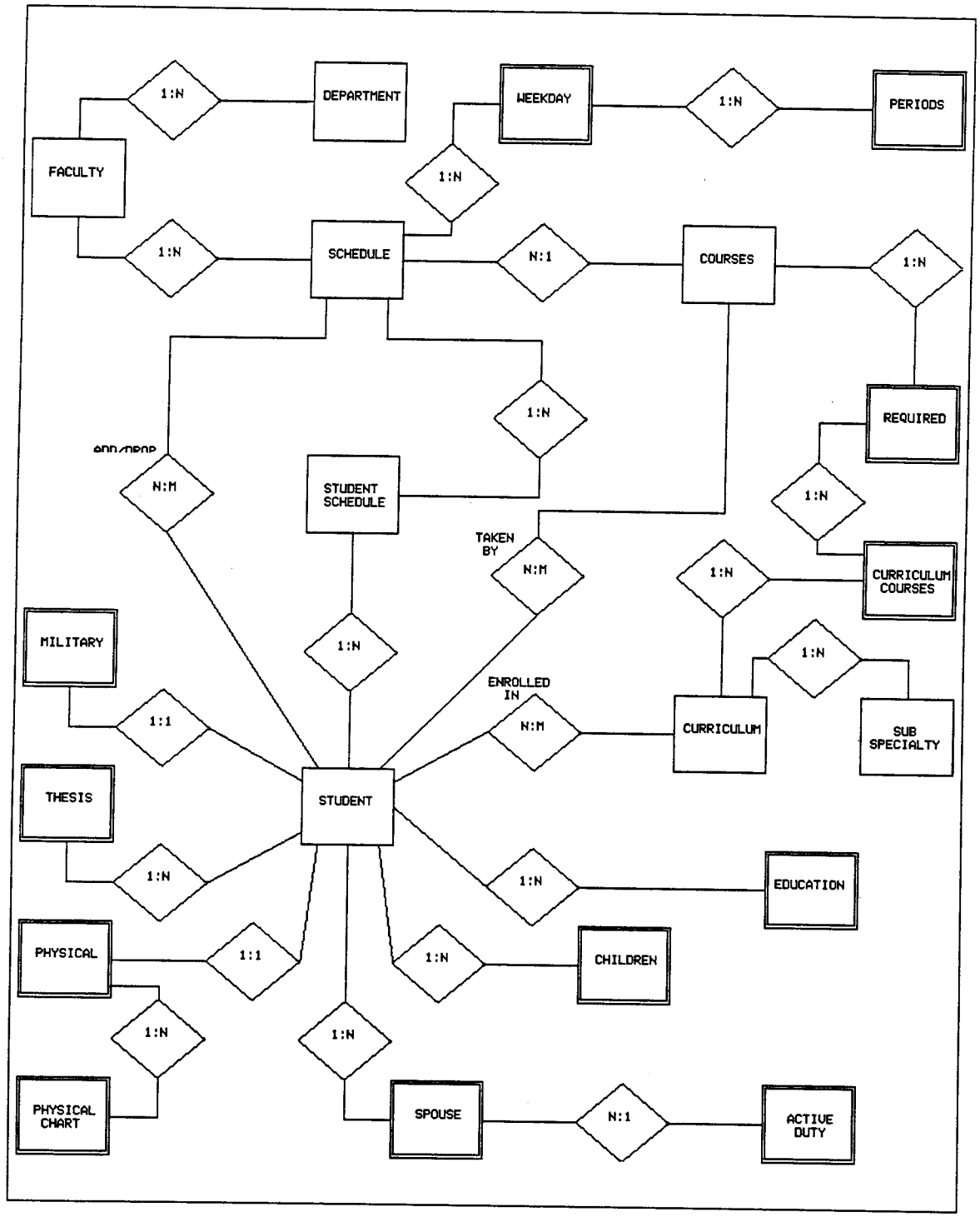
Hughes, John G., Object-Oriented Databases, Prentice Hall International (UK) Ltd, 1991.

Kroenke,Davide M., Database Processing, 4th Ed., Macmillon, 1992.

Occardi, Val, Relational Database: Theory and Practice, NCC Black Limited, 1992.

Senn, James A., Information System in Management, 4th Ed., Wadsworth, Inc., 1990.

APPENDIX A. Entity Relation Diagram



APPENDIX B. Data Dictionary

A. ENTITY DEFINITIONS

1. STUDENT Entity

- *SSN; Student-social-security-number
- LAST NAME; Student-last-name
- FIRST NAME; Student-first-name
- MIDDLE INITIAL; Student-middle-initial
- SEX; Student-sex
- MARITAL STATUS; Student-marital-status
- NUMBER OF DEPENDENTS; Student-number-of-dependents
- DOB; Student-date-of-birth
- POB; Student-place-of-birth
- SGC; Student-gard-center number
- COUNTRY; Student-Country
- STREET ADDRESS; Student-home-address
- CITY; Student-home-city
- STATE; Student-home-state
- ZIP CODE; Student-home-zip-code
- HOME PHONE; Student-home-phone
- HOME OF RECORDS; Student-home-of-record
- DATE REPORTED; Date-student-reported to NPS
- PHOTO; Student-photo

2. MILITARY Entity (Weak entity of STUDENT)

- *SSN; Student-social-security-number (foreign key)
- RANK; Student-rank
- SERVICE; Student-service

- DATE OF RANK; Student-date-of-present-rank
- SOURCE OF COMMISSION; Student-source-of-commission
- DATE OF COMMISSION; Student-date-of-commission
- YEAR GROUP; Student-year-group
- DESIGNATOR; Student-designator
- YEAR ENLISTED; Student-year-enlisted
- HIGHEST RATE; Student-highest-rate-held
- SECURITY CLEARANCE; Student-security-clearance
- QUALIFICATION; Student-qualification
- COMMUNITY; Student-community
- PAYBACK; Student-follow-on-assignment
- PREVIOUS COMMAND; Student-previous-command
- NEXT COMMAND; Student-next-command

3. **EDUCATION Entity** (Weak entity of STUDENT)
   - *SSN (foreign key)
   - COLLEGE CODE ATTENDED; College-code-attended
   - MAJOR; Description-of-course
   - LOCATION; Location-of-college-attended
   - COURSE START DATE; Course-start-date
   - COURSE END DATE; Course-end-date
   - DEGREE AWARDED; Degree-awarded

4. **SPOUSE Entity** (Weak entity of STUDENT)
   - *SPOUSE SSN; Spouse-social-security-number
   - SSN; Student-social-security-number (foreign key)
   - LAST NAME; Spouse-last-name
   - FIRST NAME; Spouse-first-name

- MIDDLE INITIAL; Spouse-middle-initial
- STREET ADDRESS; Spouse-home-address
- CITY; Spouse-home-city
- STATE; Spouse-home-state
- ZIP CODE; Spouse-home-zip-code
- ACTIVE DUTY; Spouse-active-duty

5. **SPOUSE ACTIVE DUTY Entity** (Weak entity of SPOUSE)
   - *SPOUSE SSN; Spouse-social-security-number (Foreign key)
   - SP RANK; Spouse-rank
   - SERVICE; Spouse-branch-of-service
   - STATION; Spouse-present-duty-station

6. **CHILDREN Entity** (Weak entity of STUDENT)
   - *SSN; Student-SSN (Foreign key)
   - LAST NAME; Child-last-name
   - FIRST NAME; Child-first-name
   - MIDDLE INITIAL; Child-middle-initial
   - DOB; Child-data-of-birth
   - GENDER; Child-sex

7. **FACULTY Entity**
   - *FACULTY ID; Faculty-id-code
   - LAST NAME; Faculty-last-name
   - FIRST NAME; Faculty-first-name
   - MIDDLE INITIAL; Faculty-middle-initial
   - DEPT CODE; Faculty-department-code
   - ROOM; Faculty-room-number
   - TEL; Faculty-telephone-number

- EMAIL; Faculty-email-address

8. **COURSES Entity**
   - *COURSE NO; Course-number
   - COURSE TITLE; Course-title
   - CREDIT; Credit-hours
   - LAB; Laboratory-hours

9. **SCHEDULE Entity**
   - *COURSE NO; Course-number (Foreign key)
   - *SEGMENT NO; Segment-Number
   - FACULTY ID; Faculty-id-code
   - ROOM; Room-course-taught-in

10. **WEEKDAY Entity** (Weak entity of SCHEDULE)
    - *COURSE NO; Course-number (Foreign key)
    - *SEGMENT NO; Segment-Number
    - TDAY; Date-course-offered
    - PERIOD; period-offered

11. **CURRICULUM Entity**
    - *CURR NO; Curricular-number
    - CURR TITLE; Curricular-title
    - CREDIT 4L; Credit-required-at-4000-level
    - CREDIT 3L; Credit-required-at-3000-level
    - QUARTERS; Number-of-quarters-required
    - ACADEMIC CODE; Academic-associate-code
    - P CODE; Subspecialty-code

- OFFICE CODE; Curricular-office code

12. **THESIS Entity** (Weak entity of STUDENT)
    - *SSN; Student-SSN (Foreign key)
    - TOPIC; Thesis-topic
    - ACADEMIC; Thesis-academic-associate
    - ADVISOR; Thesis-co-advisor
    - CONUS; CONUS
    - DIP STREET; Diploma-street
    - DIP CITY; Diploma-city
    - DIP STATE; Diploma-state
    - DIP ZIP; Diploma-zip-code

13. **DEPARTMENT Entity**
    - *DEPT CODE; Department-code
    - DEPT NAME; Department-name

14. **TAKEN BY Relation**
    - *SSN; Student-SSN (Foreign key)
    - *COURSE NO; Course-number (Foreign key)
    - *QUARTER ORDER; Curriculum-quarter-order
    - GRADE; Student-grade
    - TYPE; Course-type

15. **ENROLLED IN Relation**
    - *SSN; Student-SSN (Foreign key)
    - *CURR NO; Curriculum-number (Foreign key)
    - *SECTION NO; Section-number

- DATE ENROLLED; Date-enrolled
- GRADUATION DATE; Graduation-date
- CARREL; Student-study-space-no

16. **ADD/DROP Relation**
    - *SSN; Student-SSN (Foreign key)
    - *COURSE NO; Course-number (Foreign key)
    - *SEGMENT NO; Segment-Number
    - DATE; Date-of-transaction
    - TYPE; Transaction-type

17. **PHYSICAL Entity** (Weak entity of STUDENT)
    - *SSN; Student-SSN (Foreign key)
    - EXAM DATE; Date-last-physical-readiness-test
    - NEXT EXAM; Date-next-physical-test
    - HEIGHT; Student-height
    - WEIGHT; Student-weight
    - NECK; Student-neck-size
    - ABDOMEN;
    - WAIST;
    - HIP;
    - BODYFAT; Bodyfat-percentage-student
    - RESULT;
    - SIT REACH;
    - CURL UPS;
    - PUSH UPS
    - SWIM;
    - CLASSIFICATION;

18. STUDENT SCHEDULE Entity

- *SSN; Student-SSN (Foreign key)
- *COURSE NO; Course-number (Foreign key)
- *SEGMENT NO; Segment-Number
- QUARTER ORDER; Curriculum-quarter-order
- S TYPE; Course-type

19. CURRICULUM COURSES Entity

- *CURR NO; Curriculum-number (Foreign key)
- *ORDER; Curriculum-quarter-order

20. REQUIRED Entity

- *CURR NO; Curriculum-number (Foreign key)
- *ORDER; Curriculum-quarter-order (Foreign key)
- *COURSE NO; Course-number (Foreign key)
- TYPE; Course-type

21. SUBSPECIALTY Entity

- *P CODE; Subspecialty-code
- SUBSPECIALTY; Curriculum-subspecialty-title

B. DOMAIN DEFINITIONS

- Social-security-number
  - Alphanumeric 9
  - Social security number of service student

- Student-last-name
  - Text 15
  - Last name of service student

- Student-first-name
  - Text 15
  - First name of service student

- Student-middle-initial
  - Text 1
  - Middle initial of service student

- Student-sex
  - Text 1, Mask M or F
  - Gender

- Student-marital-status
  - Text, Mask M or S or D
  - Marital status of student, married, single, or divorced

- Student-number-of-dependents
  - Numeric 2
  - Number of dependents

- Student-date-of-birth
  - Date, Mask Da/Mo/Yr
  - Date of members birth

- Student-place-of-birth
  - Text 20
  - Place where the student is born

- Student-gard-center-number
  - Numeric 4
  - Student mail box number

- Student-country
  - Text 15
  - Student original country

- Student-street-address
  - Text 15
  - Student street address

- Student-home-city
  - Text 10
  - Student home city

- Student-home-state
  - Text 2, Mask XX, where XX two letter state abbreviation
  - Student home state

- Student-home-zip-code
  - Text 10, Mask XXXXX-XXXX, where X any number
  - Student home zip code

- Student-home-phone
  - Text 12, Mask XXX-XXX-XXXX, where X any number
  - Student home phone

- Student-home-of-record
  - Text 15
  - Student home city, state home of record

- Date student-reported
  - Date, Mask da/mo/yr
  - Date student reported for duty to NPS

- Student-photo
  - Image
  - Holds the students photograph

- Student-rank
  - Text 5, Mask ENS, LTJG, LT, LCDR, CDR, CAPT, RADM, VADM, ADM, LT,1LT, CPT, MAJ, LTCOL, COL, BGEN, MGEN, GEN
  - Rank of officers, standard Military abbreviation

- Student-date-of-rank
  - Date, Mask da/mo/yr
  - Student date of last promotion to current rank

- Student-service
  - Text 4, Mask USN, USA, USMC, INTL
  - Student service

- Student-source-of-commission
  - Text 7, Mask ACAD, OCS, NROTC, DIRECT
  - Student source of commissioning, std Military abbreviation

- Student-date-of-commission
  - Date, Mask da/mo/yr
  - Date student commissioned

- Student-year-group
  - Text 2
  - Student year group

- Student-designator
  - Text 4
  - Student designator

- Student-year-enlisted
  - Numeric 2

- Student-highest-rate-held
  - Text 2, Mask E1 -E9

- Student-security-clearance
  - Text 2, Mask N, C, S, TS
  - Clearance level student held

- Student-qualification
  - Text 20

- Student-community
  - Text 15

- Student-P-code
  - Text 5
  - Student subspecialty code

- Student-payback
  - Text 3, Mask YES, NO

- Student-previous-command
  - Text 15
  - Student previous command or ship and homeport

- Student-next-command
  - Text 15
  - Student next command or ship and homeport

- College-code-attended
  - Text 25
  - University or collage for undergrad degree

- Description-of-course
  - Text 15
  - Material that covered under this course

- Location-of-college-attended
  - Text 15
  - Location of the undergrad collage

- Course-start-date
  - Date, Mask da/mo/yr
  - Date started undergrade

- Course-end-date
  - Date, Mask da/mo/yr
  - Date finished undergrad

- Degree-awarded
  - Text 15
  - Title of undergrad degree

- Spouse-social-security-number
  - Alphanumeric 11
  - Social security number of the service members spouse

- Spouse-last-name
  - Text 15
  - Last name of spouse

- Spouse-first-name
  - Text 15
  - First name of spouse

- Spouse-middle-initial
  - Text 1
  - Middle initial of spouse

- Spouse-home-address
  - Text 30
  - Spouse street address
- Spouse-home-city
  - Text 20
  - Spouse home city

- Spouse-home-state
  - Text 2
  - Spouse home state

- Spouse-home-zip-code
  - Text 10, Mask XXXXX-XXXX, where X any number
  - Spouse home zip code

- Spouse-active-duty
  - Text 3, Mask YES or NO

- Spouse-rank
  - Text 5, Mask ENS, LTJG, LT, LCDR, CDR, CAPT, RADM, VADM, ADM, 1LT, LT, CPT, MAJ, LTCOL, COL, BGEN, MGEN, GEN
  - Spouse rank , standard Military abbreviation

- Spouse-branch-of-service
  - Text 20
  - Spouse branch of service

- Spouse-present-duty-station
  - Text 20
  - Spouse present duty station location

- Child-last-name
  - Text 15
  - Last name of service members child

- Child-first-name
  - Text 15
  - First name of service members child

- Child-middle-initial
  - Text 1
  - Middle initial of service members child

- Child-date-of-birth
  - Date, Mask da/mo/yr
  - Date of birth (service member child)

- Child-sex
  - Text 1, Mask M or F
  - Child sex

- Faculty-id-code
  - Text 2
  - Identification code of faculty member

- Faculty-last-name
  - Text 15
  - Last name of faculty member

- Faculty-first-name
  - Text 15
  - First name of faculty member

- Faculty-middle-initial
  - Text 1
  - Middle initial of faculty member

- Faculty-department-code
  - Text 3
  - Department code of faculty member

- Faculty-room-number
  - Text 4
  - Office number of faculty member

- Faculty-telephone-number
  - Text 13, Mask XXX-XXX-XXXX, Where X any number
  - Office phone number of faculty member

- Faculty-email-address
  - Text 15
  - Faculty email address

- Course-number
  - Text 7, Mask XXYYYY, where X any letter, Y any number
  - Course number offered

- Course-title
  - Text 40
  - Title of course being taken

- Credit-hours
  - Numeric 2, Mask XX, where X any number
  - Credit hours for course offered

- Laboratory-hours
  - Numeric 2, Mask XX, where X any number
  - Laboratory hours for course offered

- Segment-Number
  - Text 2, Mask XX, where XX any number
  - Class section number

- Room-course-taught-in
  - Text 5
  - Classroom class taught-in

- Date-course-offered
  - Text 2, Mask Mo, Tu, We, Th or Fr
  - Date class meets

- Period-offered
  - Text 1, Mask 1, 2, 3, 4, 5, 6, 7 or 8
  - Period class meets

- Curricular-number
  - Text 7
  - Curriculum number

- Curricular-title
  - Text 40
  - Curriculum description

- Credit-required-at-4000-level
  - Numeric 2
  - Credit hours required at the 4000 level

- Credit-required-at-3000-level
  - Numeric 2
  - Credit hours required at the 3000 level

- Number-of-quarters-required
  - Numeric 2
  - Number of quarters required

- Academic-associate-code
  - Text 2
  - Code of academic associate

- Subspeciality-code
  - Text 5
  - Subspecialty code of service member

- Curricular-office code
  - Text 2
  - Curricular office code

- Thesis-topic
  - Text 25
  - Topic for thesis

- Thesis-academic-associate
  - Text 15
  - Name of the academic associate

- Thesis-advisor
  - Text 15
  - Name of thesis Co advisor

- CONUS
  - Text 1, Mask Y or N

- Diploma-street
  - Text 15
  - Street address for diploma mailing

- Diploma-city
    - Text 15
    - City address for diploma mailing

- Diploma-state
    - Text 2, Mask XX, where X any letter
    - State abbreviation for diploma mailing

- Diploma-zip-code
    - Text 10, Mask XXXXX-XXXX, where X any number
    - Diploma zip code

Department-code
    - Text 3
    - Department identification code

- Department-name
    - Text 30
    - Title of department

- Course-order
    - Text 2

- Course-type
    - Text 1, Mask R, E, T or V

- Curriculum-quarter-order
    - Text 2

- Student-grade
  - Text 2, Mask A^+, A, A^-, B^+, B, B^-, C^+, C, C^-, D^+, D, D^-, X, I, W, P, F, T
  - Student grade

- Date-enrolled
  - Date, Mask da/mo/yr
  - Date student enrolled at NPS

- Graduation-date
  - Date. Mask da/mo/yr
  - Date of expected graduation

- Student-study-space-no
  - Text 3
  - Study carrel of study area

- Date-of-transaction
  - Date, Mask da/mo/yr
  - Date of transaction occurred

- Transaction-type
  - Text 1, mask Add or Drop
  - Type of transaction either add or drop

- Date-last-physical-readiness-test
  - Date, Mask da/mo/yr
  - Last PRT taken

- Date-next-physical-test
  - Date, Mask da/mo/yr
  - next PRT to be taken

- Student-height
  - Numeric 4, Mask XX.X, where X any number
  - Height in inches

- Student-weight
  - Numeric 5, Mask XXX.X, where X any number
  - Weight in pound

- Student-neck-size
  - Numeric 4, Mask XX.X, where X any number
  - Neck circumference in inches

- Student-abdomen
  - Numeric 4, Mask XX.X, where X any number
  - Abdomen circumference in inches (female)

- Student-waist
  - Numeric 4, Mask XX.X, where X any number
  - Waist circumference in inches (male)

- Student-hip
  - Numeric 4, Mask XX.X, where X any number
  - Hip circumference in inches (female)

- Bodyfat-percentage-student
  - Numeric 2, Mask XX, where X any number

- Result
  - Numeric 5, Mask XXX.X, where X any number
  - Percent body fat using DOD scale

- Student-sit-reach
    - Text 1, P or F

- Student-curl-ups
    - Numeric 3, Mask XXX, where X any number
    - Number of sit-ups achieved

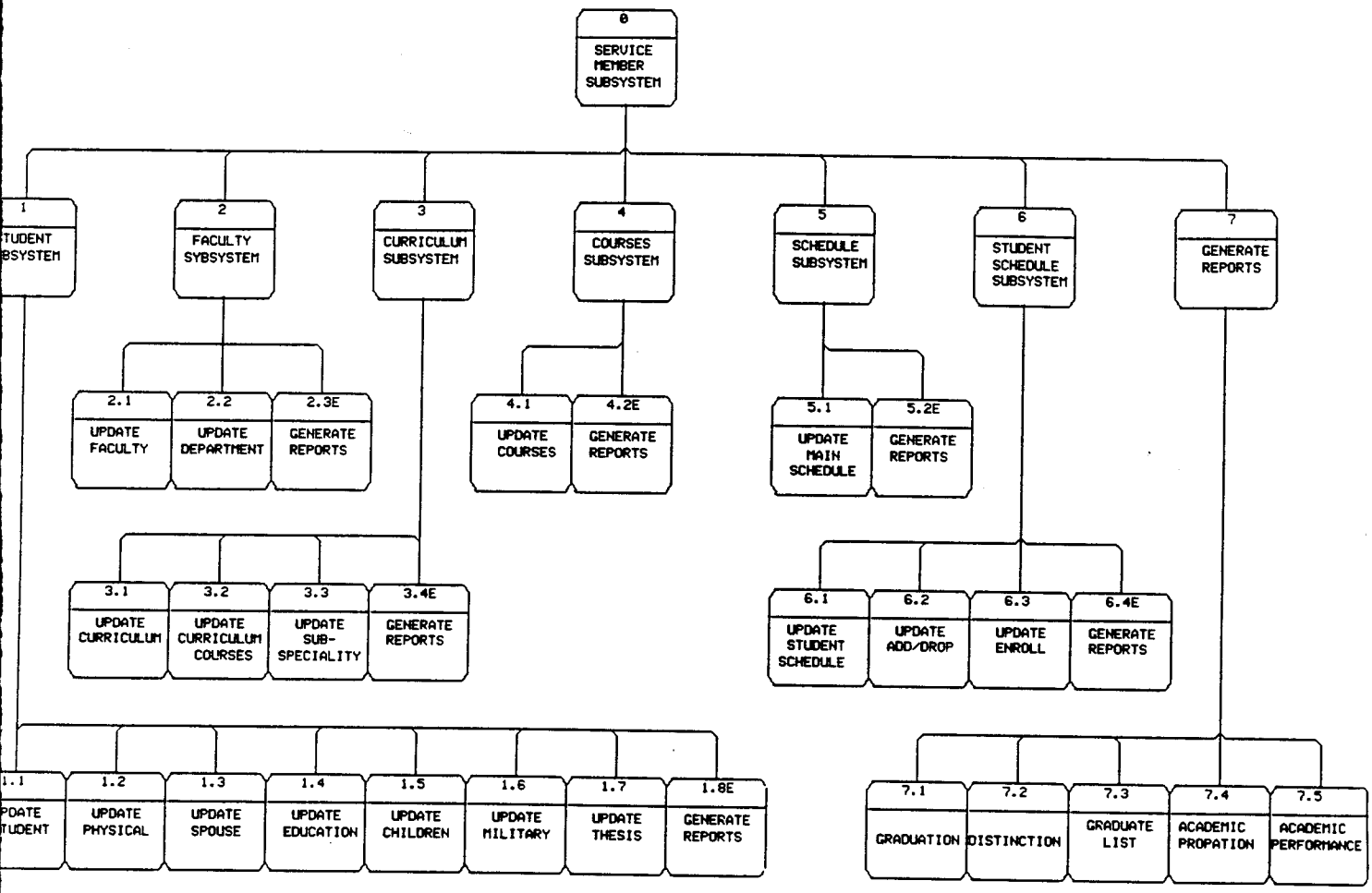
- Student-push-ups
    - Numeric 3, Mask XXX, where X any number
    - Number of pushups achieved

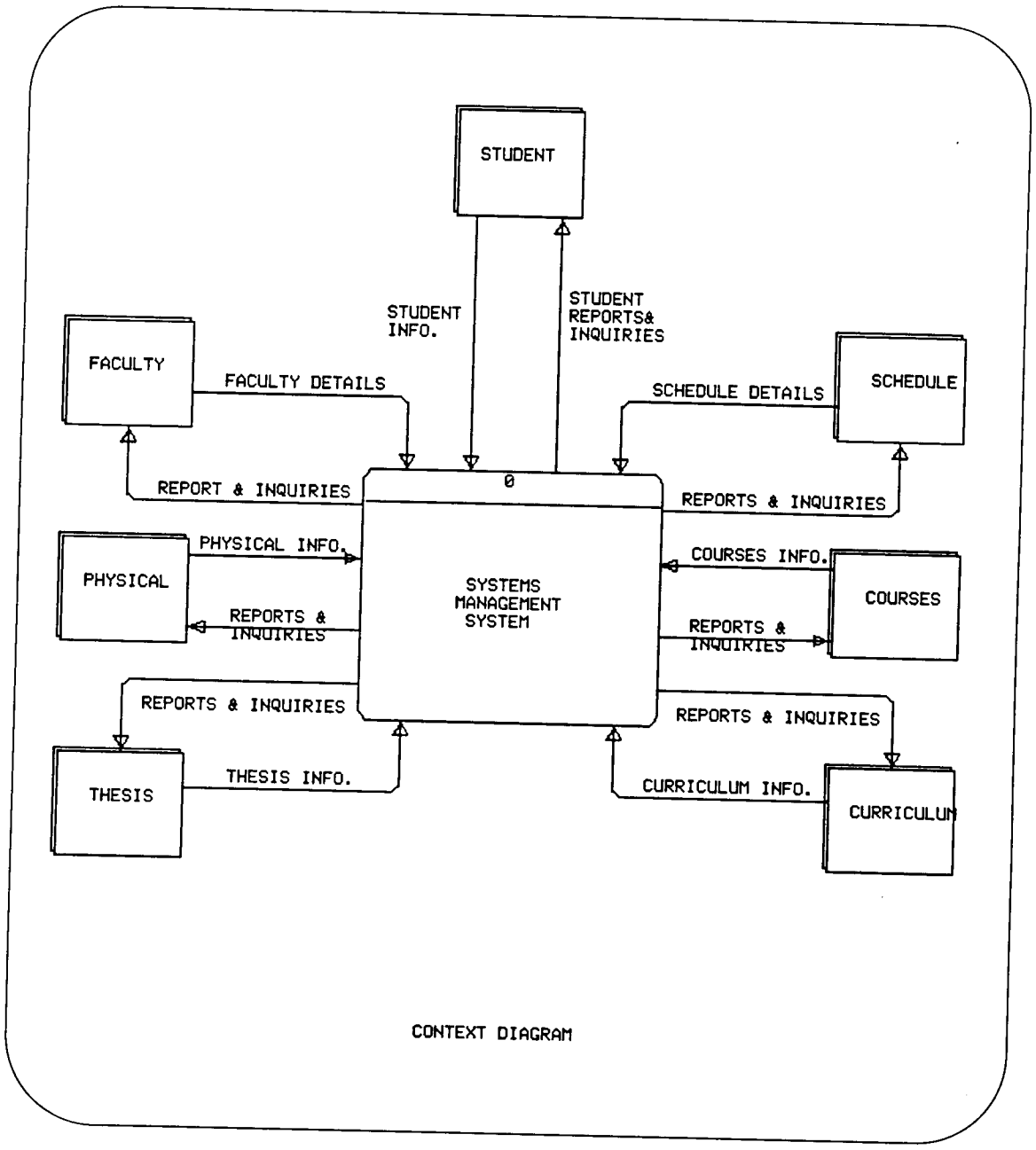
- Student-swim
    - Numeric 5, Mask XX:XX, where X any number
    - Time for 500 yd swim

- Student-classification
    - Text 15, Mask Outstanding, Excellent, Good or Satisfactory
    - Overall score for PRT

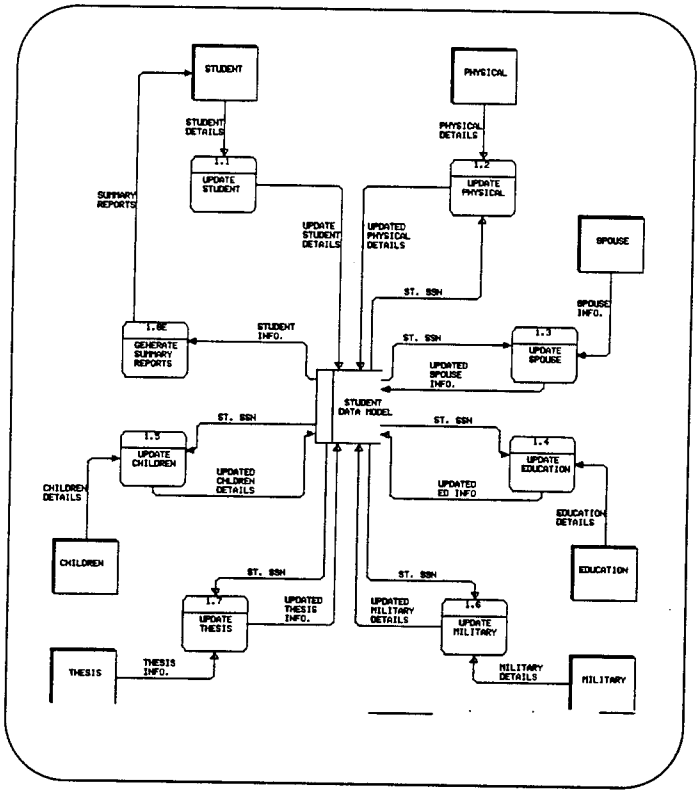
- Curriculum-subspecialty-title
    - Text 30
    - Subspecialty title

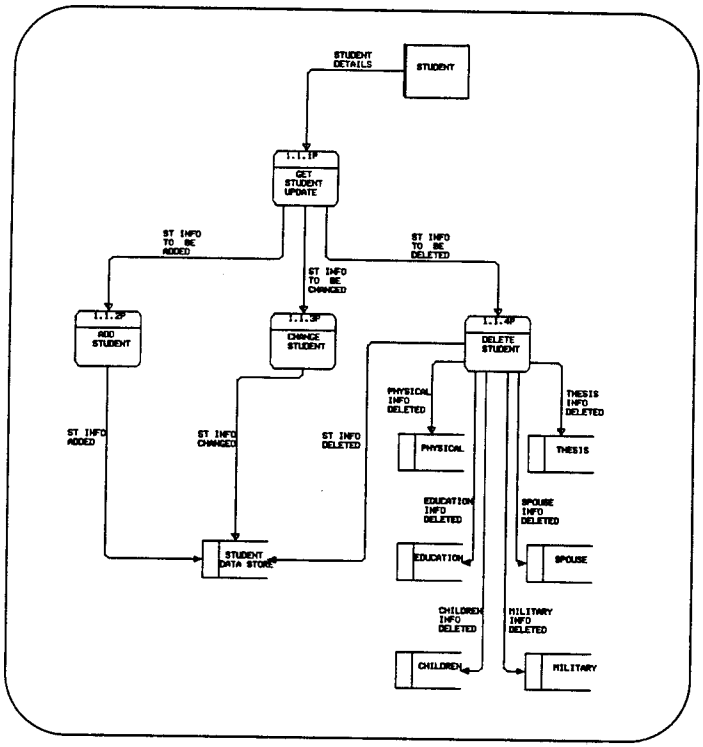
APPENDIX C. Data Flow Diagrams

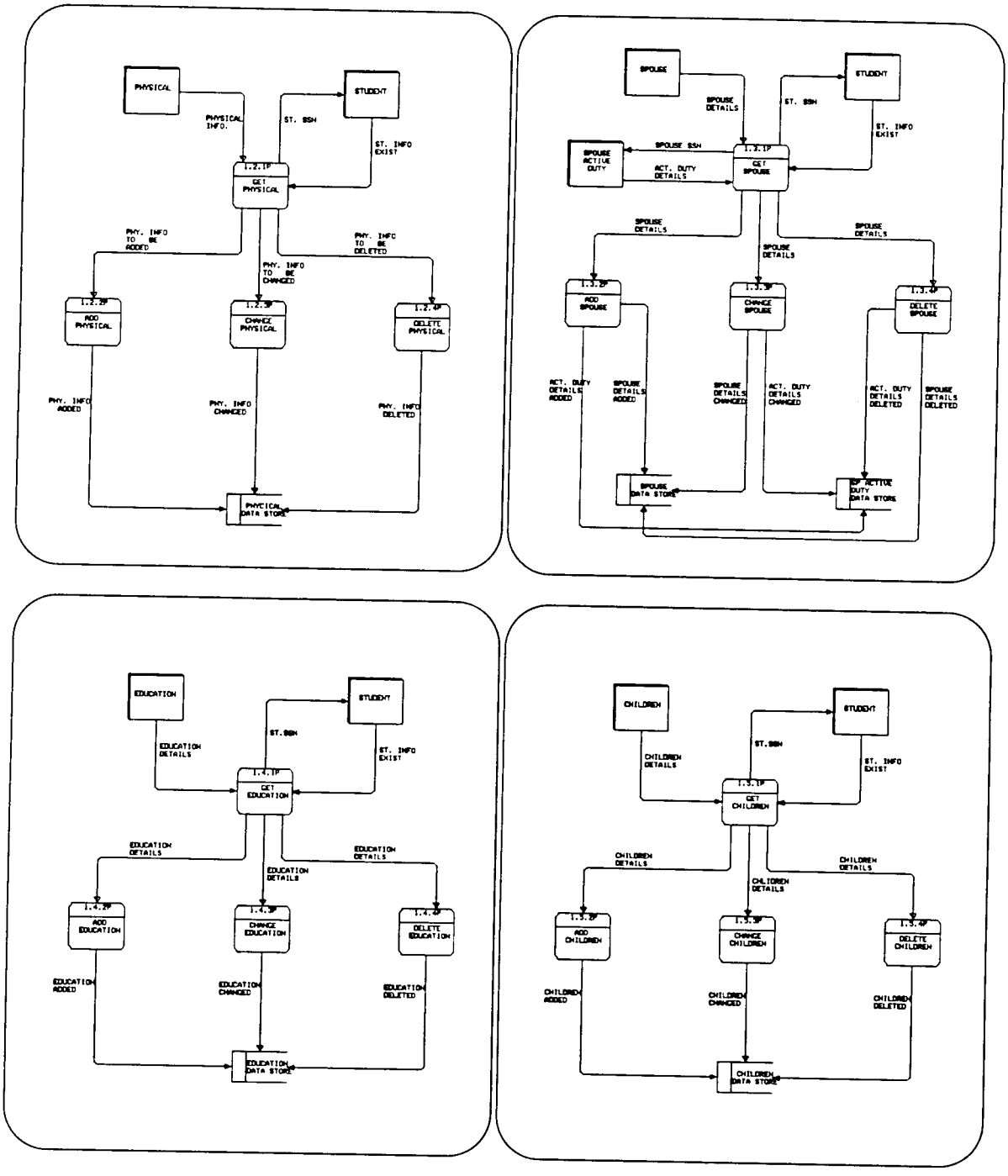


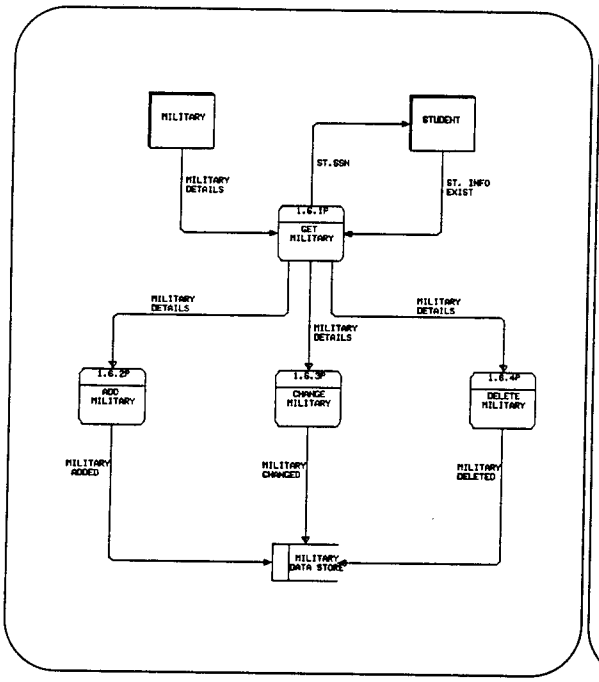


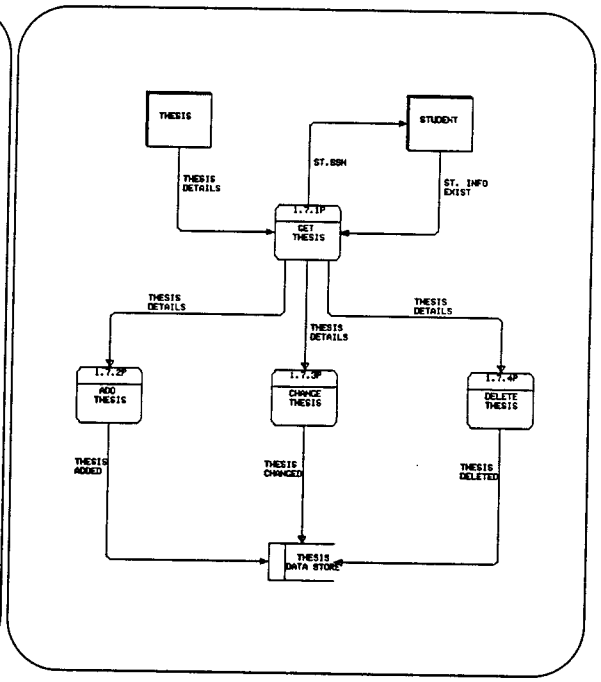
CONTEXT DIAGRAM

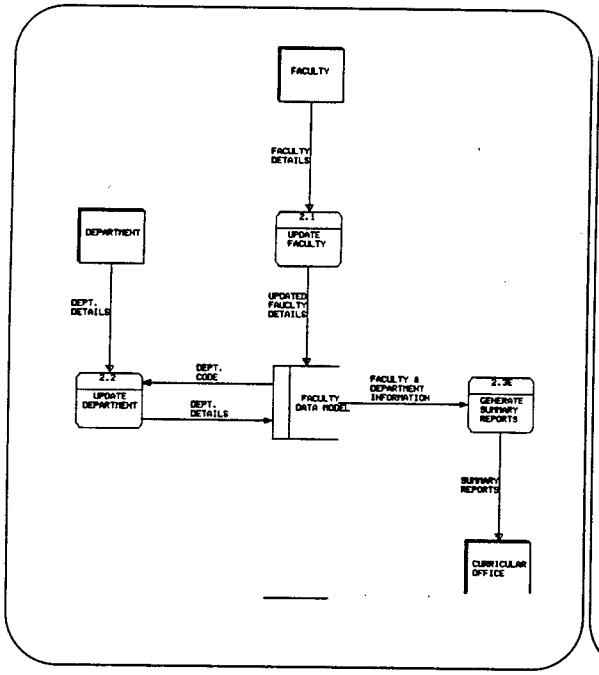

STUDENT
PHYSICAL
STUDENT DETAILS
PHYSICAL DETAILS
1.1 UPDATE STUDENT
1.2 UPDATE PHYSICAL
SUMMARY REPORTS
UPDATE STUDENT DETAILS
UPDATED PHYSICAL DETAILS
SPOUSE
ST. SSN
SPOUSE INFO.
1.8E GENERATE SUMMARY REPORTS
STUDENT INFO.
ST. SSN
1.3 UPDATE SPOUSE
UPDATED SPOUSE INFO.
STUDENT DATA MODEL
ST. SSN
ST. SSN
1.5 UPDATE CHILDREN
1.4 UPDATE EDUCATION
CHILDREN DETAILS
UPDATED CHILDREN DETAILS
UPDATED ED INFO
EDUCATION DETAILS
CHILDREN
EDUCATION
ST. SSN
ST. SSN
1.7 UPDATE THESIS
UPDATED THESIS INFO.
UPDATED MILITARY DETAILS
1.6 UPDATE MILITARY
THESIS
THESIS INFO.
MILITARY DETAILS
MILITARY

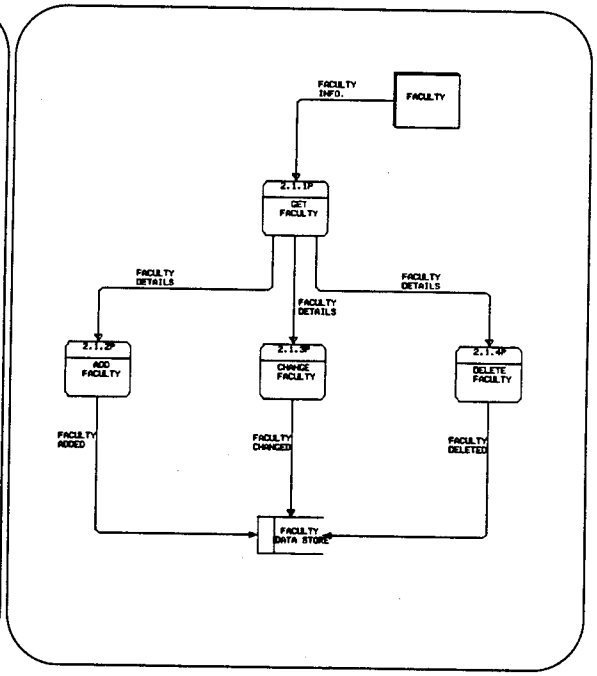

STUDENT DETAILS
STUDENT
1.1.1P GET STUDENT UPDATE
ST INFO TO BE ADDED
ST INFO TO BE CHANGED
ST INFO TO BE DELETED
1.1.2P ADD STUDENT
1.1.3P CHANGE STUDENT
1.1.4P DELETE STUDENT
ST INFO ADDED
ST INFO CHANGED
ST INFO DELETED
PHYSICAL INFO DELETED
THESIS INFO DELETED
PHYSICAL
THESIS
EDUCATION INFO DELETED
SPOUSE INFO DELETED
EDUCATION
SPOUSE
STUDENT DATA STORE
CHILDREN INFO DELETED
MILITARY INFO DELETED
CHILDREN
MILITARY

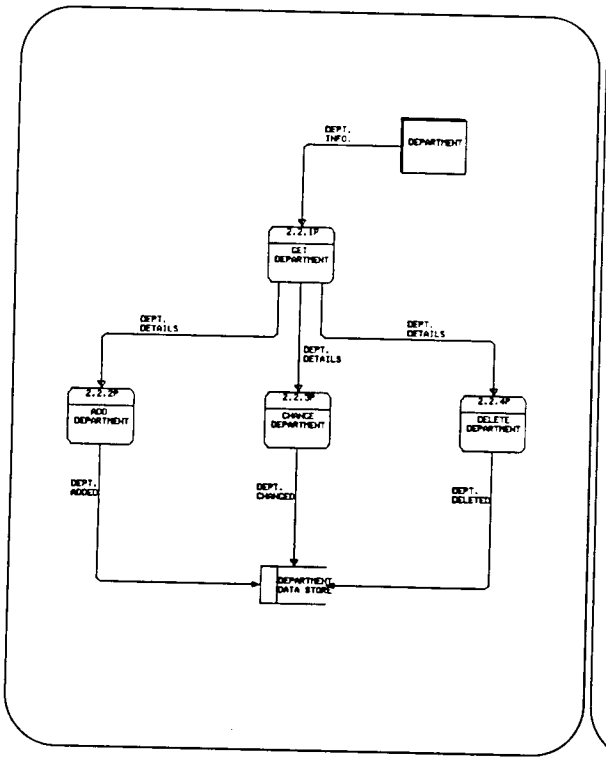

PHYSICAL
STUDENT
PHYSICAL INFO.
ST. SSN
ST. INFO EXIST
1.2.1P
GET PHYSICAL
PHY. INFO TO BE ADDED
PHY. INFO TO BE CHANGED
PHY. INFO TO BE DELETED
1.2.2P
ADD PHYSICAL
1.2.3P
CHANGE PHYSICAL
1.2.4P
DELETE PHYSICAL
PHY. INFO ADDED
PHY. INFO CHANGED
PHY. INFO DELETED
PHYCICAL DATA STORE
SPOUSE
STUDENT
SPOUSE DETAILS
ST. SSN
ST. INFO EXIST
SPOUSE ACTIVE DUTY
SPOUSE SSN
ACT. DUTY DETAILS
1.3.1P
GET SPOUSE
SPOUSE DETAILS
SPOUSE DETAILS
SPOUSE DETAILS
1.3.2P
ADD SPOUSE
1.3.3P
CHANGE SPOUSE
1.3.4P
DELETE SPOUSE
ACT. DUTY DETAILS ADDED
SPOUSE DETAILS ADDED
SPOUSE DETAILS CHANGED
ACT. DUTY DETAILS CHANGED
ACT. DUTY DETAILS DELETED
SPOUSE DETAILS DELETED
SPOUSE DATA STORE
SP ACTIVE DUTY DATA STORE
EDUCATION
STUDENT
EDUCATION DETAILS
ST.SSN
ST. INFO EXIST
1.4.1P
GET EDUCATION
EDUCATION DETAILS
EDUCATION DETAILS
EDUCATION DETAILS
1.4.2P
ADD EDUCATION
1.4.3P
CHANGE EDUCATION
1.4.4P
DELETE EDUCATION
EDUCATION ADDED
EDUCATION CHANGED
EDUCATION DELETED
EDUCATION DATA STORE
CHILDREN
STUDENT
CHILDREN DETAILS
ST.SSN
ST. INFO EXIST
1.5.1P
GET CHILDREN
CHILDREN DETAILS
CHILDREN DETAILS
CHILDREN DETAILS
1.5.2P
ADD CHILDREN
1.5.3P
CHANGE CHILDREN
1.5.4P
DELETE CHILDREN
CHILDREN ADDED
CHILDREN CHANGED
CHILDREN DELETED
CHILDREN DATA STORE

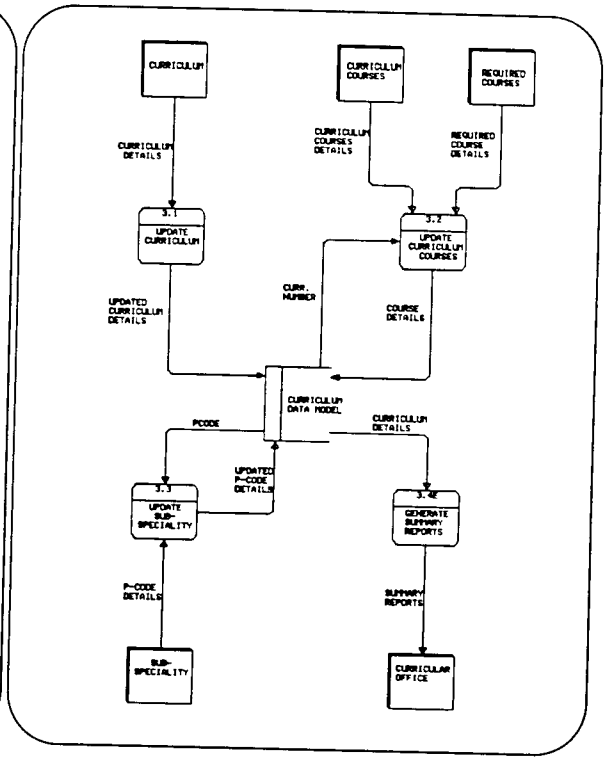

MILITARY
STUDENT
MILITARY DETAILS
ST.SSN
ST. INFO EXIST
1.6.1P
GET MILITARY
MILITARY DETAILS
MILITARY DETAILS
MILITARY DETAILS
1.6.2P
ADD MILITARY
1.6.3P
CHANGE MILITARY
1.6.4P
DELETE MILITARY
MILITARY ADDED
MILITARY CHANGED
MILITARY DELETED
MILITARY DATA STORE

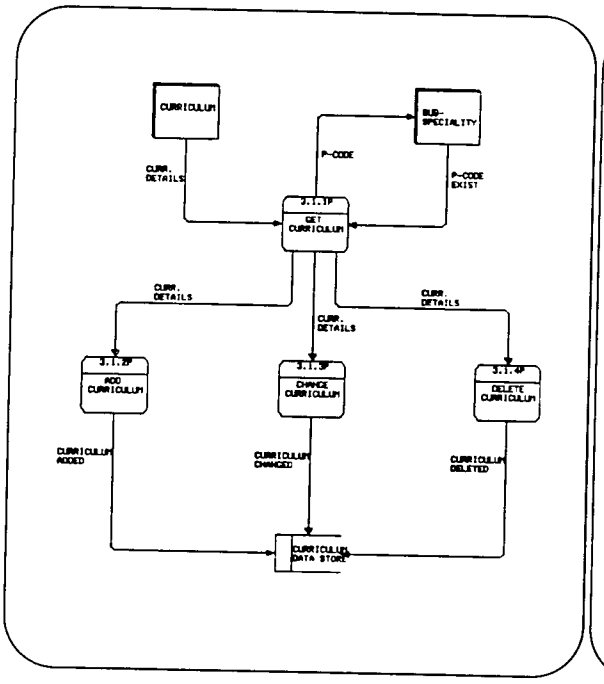

THESIS
STUDENT
THESIS DETAILS
ST.SSN
ST. INFO EXIST
1.7.1P
GET THESIS
THESIS DETAILS
THESIS DETAILS
THESIS DETAILS
1.7.2P
ADD THESIS
1.7.3P
CHANGE THESIS
1.7.4P
DELETE THESIS
THESIS ADDED
THESIS CHANGED
THESIS DELETED
THESIS DATA STORE

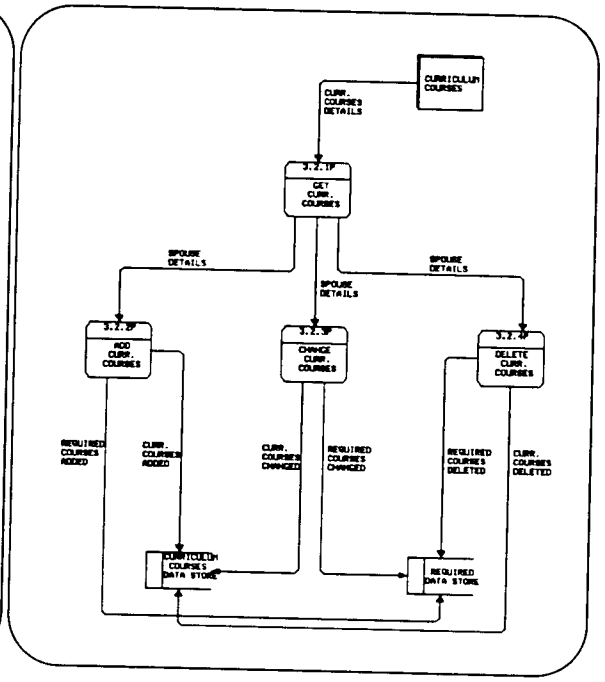

FACULTY
FACULTY DETAILS
DEPARTMENT
2.1
UPDATE FACULTY
DEPT. DETAILS
UPDATED FAUCLTY DETAILS
2.2
UPDATE DEPARTMENT
DEPT. CODE
DEPT. DETAILS
FACULTY DATA MODEL
FACULTY & DEPARTMENT INFORMATION
2.3
GENERATE SUMMARY REPORTS
SUMMARY REPORTS
CURRICULAR OFFICE

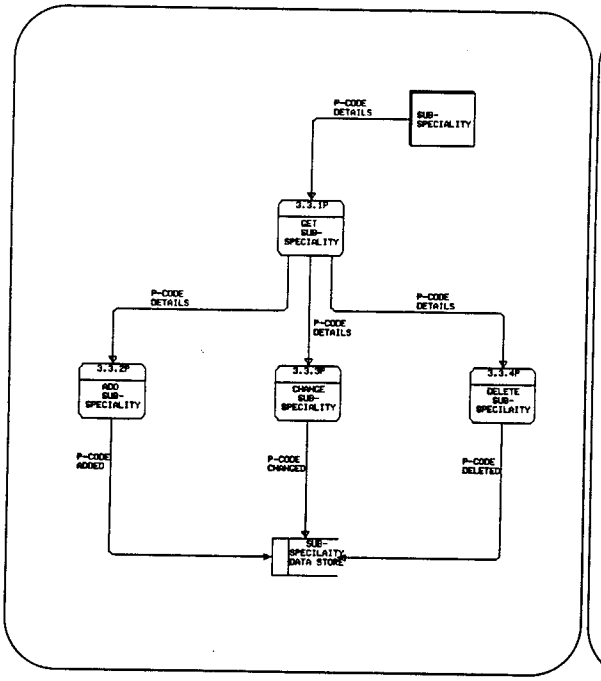

FACULTY INFO.
FACULTY
2.1.1P
GET FACULTY
FACULTY DETAILS
FACULTY DETAILS
FACULTY DETAILS
2.1.2P
ADD FACULTY
2.1.3P
CHANGE FACULTY
2.1.4P
DELETE FACULTY
FACULTY ADDED
FACULTY CHANGED
FACULTY DELETED
FACULTY DATA STORE

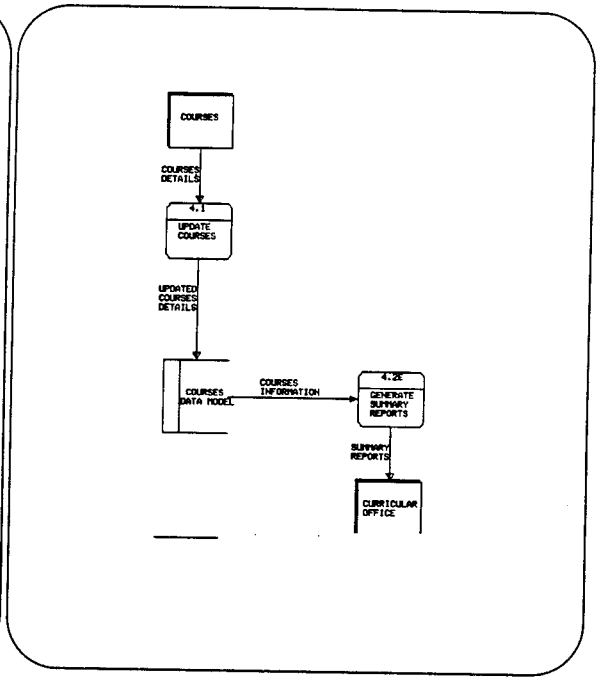

DEPT. INFO.
DEPARTMENT
2.2.1P
GET DEPARTMENT
DEPT. DETAILS
DEPT. DETAILS
DEPT. DETAILS
2.2.2P
ADD DEPARTMENT
2.2.3P
CHANGE DEPARTMENT
2.2.4P
DELETE DEPARTMENT
DEPT. ADDED
DEPT. CHANGED
DEPT. DELETED
DEPARTMENT DATA STORE

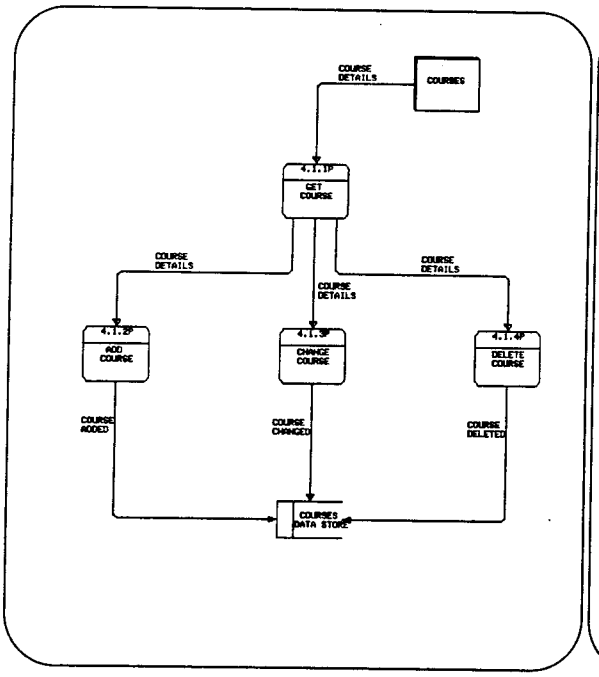

CURRICULUM
CURRICULUM COURSES
REQUIRED COURSES
CURRICULUM DETAILS
CURRICULUM COURSES DETAILS
REQUIRED COURSE DETAILS
3.1
UPDATE CURRICULUM
3.2
UPDATE CURRICULUM COURSES
UPDATED CURRICULUM DETAILS
CURR. NUMBER
COURSE DETAILS
CURRICULUM DATA MODEL
PCODE
CURRICULUM DETAILS
UPDATED P-CODE DETAILS
3.3
UPDATE SUB-SPECIALITY
3.4
GENERATE SUMMARY REPORTS
P-CODE DETAILS
SUMMARY REPORTS
SUB-SPECIALITY
CURRICULAR OFFICE

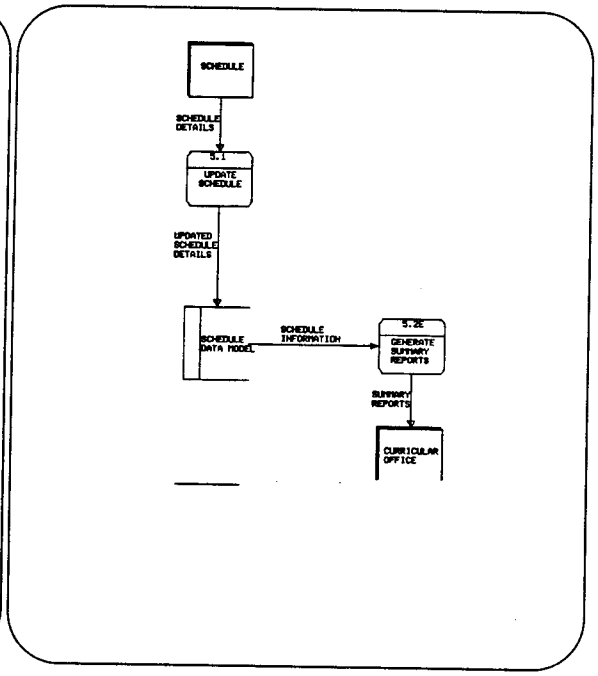

CURRICULUM
SUB-SPECIALITY
P-CODE
CURR. DETAILS
P-CODE EXIST
3.1.1P
GET CURRICULUM
CURR. DETAILS
CURR. DETAILS
CURR. DETAILS
3.1.2P
ADD CURRICULUM
3.1.3P
CHANGE CURRICULUM
3.1.4P
DELETE CURRICULUM
CURRICULUM ADDED
CURRICULUM CHANGED
CURRICULUM DELETED
CURRICULUM DATA STORE

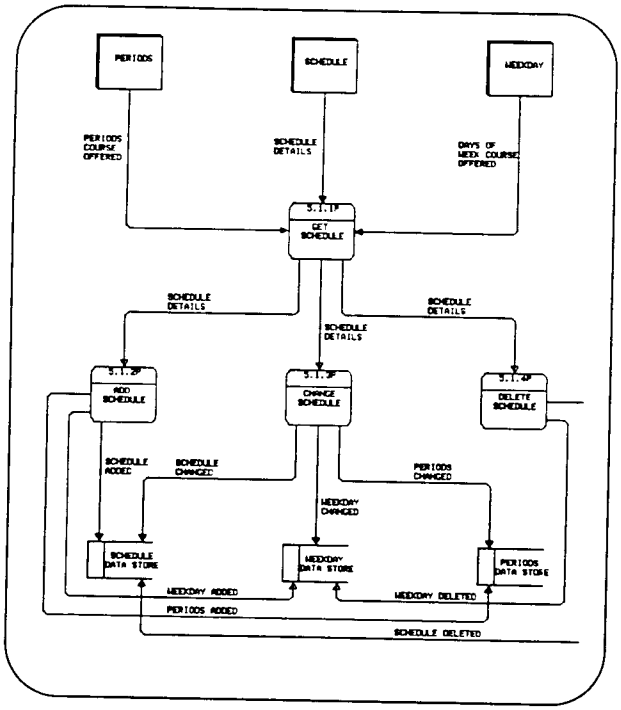

CURRICULUM COURSES
CURR. COURSES DETAILS
3.2.1P
GET CURR. COURSES
SPOUSE DETAILS
SPOUSE DETAILS
SPOUSE DETAILS
3.2.2P
ADD CURR. COURSES
3.2.3P
CHANGE CURR. COURSES
3.2.4P
DELETE CURR. COURSES
REQUIRED COURSES ADDED
CURR. COURSES ADDED
CURR. COURSES CHANGED
REQUIRED COURSES CHANGED
REQUIRED COURSES DELETED
CURR. COURSES DELETED
CURRICULUM COURSES DATA STORE
REQUIRED DATA STORE

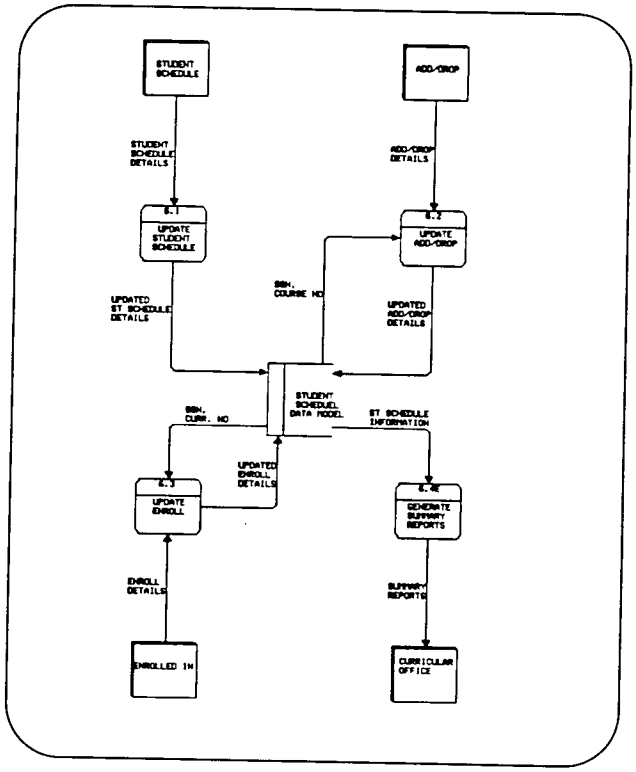

P-CODE DETAILS
SUB-SPECIALITY
3.3.1P
GET SUB-SPECIALITY
P-CODE DETAILS
P-CODE DETAILS
P-CODE DETAILS
3.3.2P
ADD SUB-SPECIALITY
3.3.3P
CHANGE SUB-SPECIALITY
3.3.4P
DELETE SUB-SPECIALITY
P-CODE ADDED
P-CODE CHANGED
P-CODE DELETED
SUB-SPECIALITY DATA STORE


COURSES
COURSES DETAILS
4.1
UPDATE COURSES
UPDATED COURSES DETAILS
COURSES DATA MODEL
COURSES INFORMATION
4.2E
GENERATE SUMMARY REPORTS
SUMMARY REPORTS
CURRICULAR OFFICE


COURSE DETAILS
COURSES
4.1.1P
GET COURSE
COURSE DETAILS
COURSE DETAILS
COURSE DETAILS
4.1.2P
ADD COURSE
4.1.3P
CHANGE COURSE
4.1.4P
DELETE COURSE
COURSE ADDED
COURSE CHANGED
COURSE DELETED
COURSES DATA STORE


SCHEDULE
SCHEDULE DETAILS
5.1
UPDATE SCHEDULE
UPDATED SCHEDULE DETAILS
SCHEDULE DATA MODEL
SCHEDULE INFORMATION
5.2E
GENERATE SUMMARY REPORTS
SUMMARY REPORTS
CURRICULAR OFFICE


PERIODS
SCHEDULE
WEEKDAY
PERIODS COURSE OFFERED
SCHEDULE DETAILS
DAYS OF WEEK COURSE OFFERED
5.1.1P
GET SCHEDULE
SCHEDULE DETAILS
SCHEDULE DETAILS
SCHEDULE DETAILS
5.1.2P
ADD SCHEDULE
5.1.3P
CHANGE SCHEDULE
5.1.4P
DELETE SCHEDULE
SCHEDULE ADDED
SCHEDULE CHANGED
PERIODS CHANGED
WEEKDAY CHANGED
SCHEDULE DATA STORE
WEEKDAY DATA STORE
PERIODS DATA STORE
WEEKDAY ADDED
WEEKDAY DELETED
PERIODS ADDED
SCHEDULE DELETED


STUDENT SCHEDULE
ADD/DROP
STUDENT SCHEDULE DETAILS
ADD/DROP DETAILS
6.1
UPDATE STUDENT SCHEDULE
6.2
UPDATE ADD/DROP
SSN, COURSE NO
UPDATED ST SCHEDULE DETAILS
UPDATED ADD/DROP DETAILS
STUDENT SCHEDULE DATA MODEL
SSN, CURR. NO
ST SCHEDULE INFORMATION
UPDATED ENROLL DETAILS
6.3
UPDATE ENROLL
6.4
GENERATE SUMMARY REPORTS
ENROLL DETAILS
SUMMARY REPORTS
ENROLLED IN
CURRICULAR OFFICE

APPENDIX D. Update, Display, and Control Mechanism

The following are attributes and functions of the Systems Management database listed in the following format:

a. Input

b. Output

c. Process notes

Note: Volume and frequency values have not been individually addressed in each process. The frequency in which a process is addressed will vary dependent on the size and structure of a given curriculum. Additionally, volume is dependent on the SM size and the number of officers and civilian assigned.

STUDENT

1. **Get Student Update (1.1.1P)**

   a. Selection from the Student menu (allows add, change, and delete service member).

   b. Update the Student data store according to selection.

   c. This process allows a choice between the different options within the Student menu.

2. **Add Student (1.1.2P)**

   a. Student information (Personnel Office).

   b. Store in Student data store.

   c. Form provided for input of each field.

3. **Change Student (1.1.3P)**

   a. Student information (Personnel/Admin Offices).

   b. Update Student data store.

   c. Queuing for each field provided after Student is found (form view).

4. **Delete Student (1.1.4P)**

   a. Student SSN (Personnel/Admin Office).

   b. Confirmation of deletion.

   c. Confirm deletion actually will remove the record from Student, and all the relating data stores.

PHYSICAL

5. **Update Physical (1.2.1P)**

   a. Select from the student menu (add, change or delete a Physical record).

   b. Update the PHYSICAL data store.

   c. This process allows a choice between the different options of the physical menu.

6. **Add Physical Details (1.2.2P)**

   a. Physical information (Disbursing Office)

   b. Store in physical data store.

   c. Form is provided for input of each field.

7. **Change Physical Details (1.2.3P)**

   a. Physical information (Disbursing Office)

   b. Update in Physical data store.

   c. Queuing by field provided after physical record found (form view).

8. **Delete Physical Details (1.2.4P)**

   a. Student SSN (Personnel/Admin Office).

   b. Confirmation of deletion.

   c. Confirm deletion actually will remove the record from Physical data stores.

SPOUSE

9. **Get Spouse Update (1.3.1P)**

    a. Selection from the spouse menu (add, change and delete).

    b. Update the spouse data store.

    c. Choice between different options of the spouse menu.

10. **Add New spouse Details (1.3.2P)**

    a. Spouse information (Admin. Office)

    b. Store in spouse data store.

    c. Form is provided for input of each field.

11. **Change Spouse Details (1.3.3P)**

    a. Spouse Information (Admin. Office).

    b. Update the Spouse data store.

    c. Queuing by field provided after spouse record found (form view).

12. **Delete spouse Details (1.3.4P)**

    a. Service Member SSN (Personnel/Admin Office).

    b. Confirmation of deletion.

    c. Confirm deletion actually desired.

CHILDREN

13. **Get Children Update Physical (1.5.1P)**

    a. Select from the children menu (add, change or delete a children record).

    b. Update the children data store.

    c. This process allows a choice between the different options of the children menu.

14. **Add Children Details (1.5.2P)**

    a. Children information (Disbursing Office)

    b. Store in children data store.

c. Form is provided for input of each field.

15. **Change Children Details (1.5.3P)**

a. Children information (Disbursing Office)

b. Update in Children data store.

c. Queuing by field provided after Children record found (form view).

16. **Delete Children Details (1.5.4P)**

a. Student SSN (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Children data stores.

MILITARY

17. **Get Military Update (1.6.1P)**

a. Select from the military menu (add, change or delete a Military record).

b. Update the PHYSICAL data store.

c. This process allows a choice between the different options of the physical menu.

18. **Add Military Details (1.6.2P)**

a. Military information (Disbursing Office)

b. Store in Military data store.

c. Form is provided for input of each field.

19. **Change Military Details (1.6.3P)**

a. Military information (Disbursing Office)

b. Update in Military data store.

c. Queuing by field provided after military record found (form view).

20. **Delete Military Details (1.6.4P)**

    a. Student SSN (Personnel/Admin Office).

    b. Confirmation of deletion.

    c. Confirm deletion actually will remove the record from Military data stores.

THESIS

21. **Get Thesis Update (1.7.1P)**

    a. Select from the Thesis menu (add, change or delete a Thesis record).

    b. Update the Thesis data store.

    c. This process allows a choice between the different options of the Thesis menu.

22. **Add Physical Details (1.7.2P)**

    a. Thesis information (Disbursing Office)

    b. Store in thesis data store.

    c. Form is provided for input of each field.

23. **Change Physical Details (1.7.3P)**

    a. Thesis information (Disbursing Office)

    b. Update in Thesis data store.

    c. Queuing by field provided after Thesis record found (form view).

24. **Delete Physical Details (1.7.4P)**

    a. Student SSN (Personnel/Admin Office).

    b. Confirmation of deletion.

    c. Confirm deletion actually will remove the record from Thesis data stores.

FACULTY

25. **Get Faculty Update (2.1.1P)**

    a. Select from the faculty menu (add, change or delete a Faculty record).

    b. Update the Faculty data store.

c. This process allows a choice between the different options of the Faculty menu.

26. **Add Faculty Details (2.1.2P)**

a. Faculty information (Disbursing Office)

b. Store in faculty data store.

c. Form is provided for input of each field.

27. **Change Faculty Details (2.1.3P)**

a. Faculty information (Disbursing Office)

b. Update in Faculty data store.

c. Queuing by field provided after faculty record found (form view).

28. **Delete Faculty Details (2.1.4P)**

a. Faculty ID.(Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Faculty data stores.

DEPARTMENT

29. **Get Department Update (2.2.1P)**

a. Select from the department menu (add, change or delete a department record).

b. Update the Department data store.

c. This process allows a choice between the different options of the Department menu.

30. **Add Department Details (2.2.2P)**

a. Department information (Disbursing Office)

b. Store in Department data store.

c. Form is provided for input of each field.

31. **Change Department Details (2.2.3P)**

a. Department information (Disbursing Office)

b. Update in Department data store.

c. Queuing by field provided after Department record found (form view).

32. **Delete Department Details (2.2.4P)**

a. Department code (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Department data stores.

CURRICULUM

33. **Get Curriculum Update (3.1.1P)**

a. Select from the Curriculum menu (add, change or delete a curriculum record).

b. Update the Curriculum data store.

c. This process allows a choice between the different options of the Curriculum menu.

34. **Add Curriculum Details (3.1.2P)**

a. curriculum information (Disbursing Office)

b. Store in Curriculum data store.

c. Form is provided for input of each field.

35. **Change Curriculum Details (3.1.3P)**

a. Curriculum information (Disbursing Office)

b. Update in Curriculum data store.

c. Queuing by field provided after Curriculum record found (form view).

36. **Delete Curriculum Details (3.1.4P)**

a. Curriculum Number (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Curriculum data stores.

Curriculum Courses

37. **Get Curriculum Courses Update (3.2.1P)**

a. Select from the curriculum courses menu (add, change or delete a record).

b. Update the curriculum courses and Required data store.

c. This process allows a choice between the different options of the curriculum courses menu.

38. **Add Curriculum Courses Details (3.2.2P)**

a. Curriculum Courses and Required information (Disbursing Office)

b. Store in curriculum courses and required data store.

c. Form is provided for input of each field.

39. **Change Curriculum Courses Details (3.2.3P)**

a. Curriculum Courses and Required information (Disbursing Office)

b. Update in curriculum courses and required data store.

c. Queuing by field provided after Curriculum courses record found (form view).

40. **Delete Curriculum Courses Details (3.2.4P)**

a. Course Number (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Curriculum Courses and Required data stores.

SUBSPECIALTY

41. **Get Subspecialty Update (3.3.1P)**

a. Select from the subspecialty menu (add, change or delete subspecialty record).

b. Update the subspecialty data store.

c. This process allows a choice between the different options of the subspecialty menu.

42. **Add Subspecialty Details (3.3.2P)**

a. Subspecialty information (Disbursing Office)

b. Store in Subspecialty data store.

c. Form is provided for input of each field.

43. **Change Subspecialty Details (3.3.3P)**

a. Subspecialty information (Disbursing Office)

b. Update in Subspecialty data store.

c. Queuing by field provided after Subspecialty record found (form view).

44. **Delete Subspecialty Details (3.3.4P)**

a. Subspecialty code (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Subspecialty data stores.

COURSES

45. **Get Courses Update (4.1.1P)**

a. Select from the courses menu (add, change or delete a courses record).

b. Update the Courses data store.

c. This process allows a choice between the different options of the courses menu.

46. **Add Courses Details (4.1.2P)**

a. Courses information (Disbursing Office)

b. Store in courses data store.

c. Form is provided for input of each field.

47. **Change Courses Details (4.1.3P)**

a. Courses information (Disbursing Office)

b. Update in Courses data store.

c. Queuing by field provided after Courses record found (form view).

48. **Delete Courses Details (4.1.4P)**

a. Course number (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Courses data stores.

SCHEDULE

49. **Get Schedule Update (5.1.1P)**

a. Select from the schedule menu (add, change or delete a schedule record).

b. Update the Schedule, Weekday and Periods data store.

c. This process allows a choice between the different options of the schedule menu.

50. **Add Schedule Details (5.1.2P)**

a. Schedule information (Disbursing Office)

b. Store in Schedule, Weekday and Periods data store.

c. Form is provided for input of each field.

51. **Change Courses Details (5.1.3P)**

a. Schedule information (Disbursing Office)

b. Update in Schedule, Weekday and Periods data store.

c. Queuing by field provided after Schedule record found (form view).

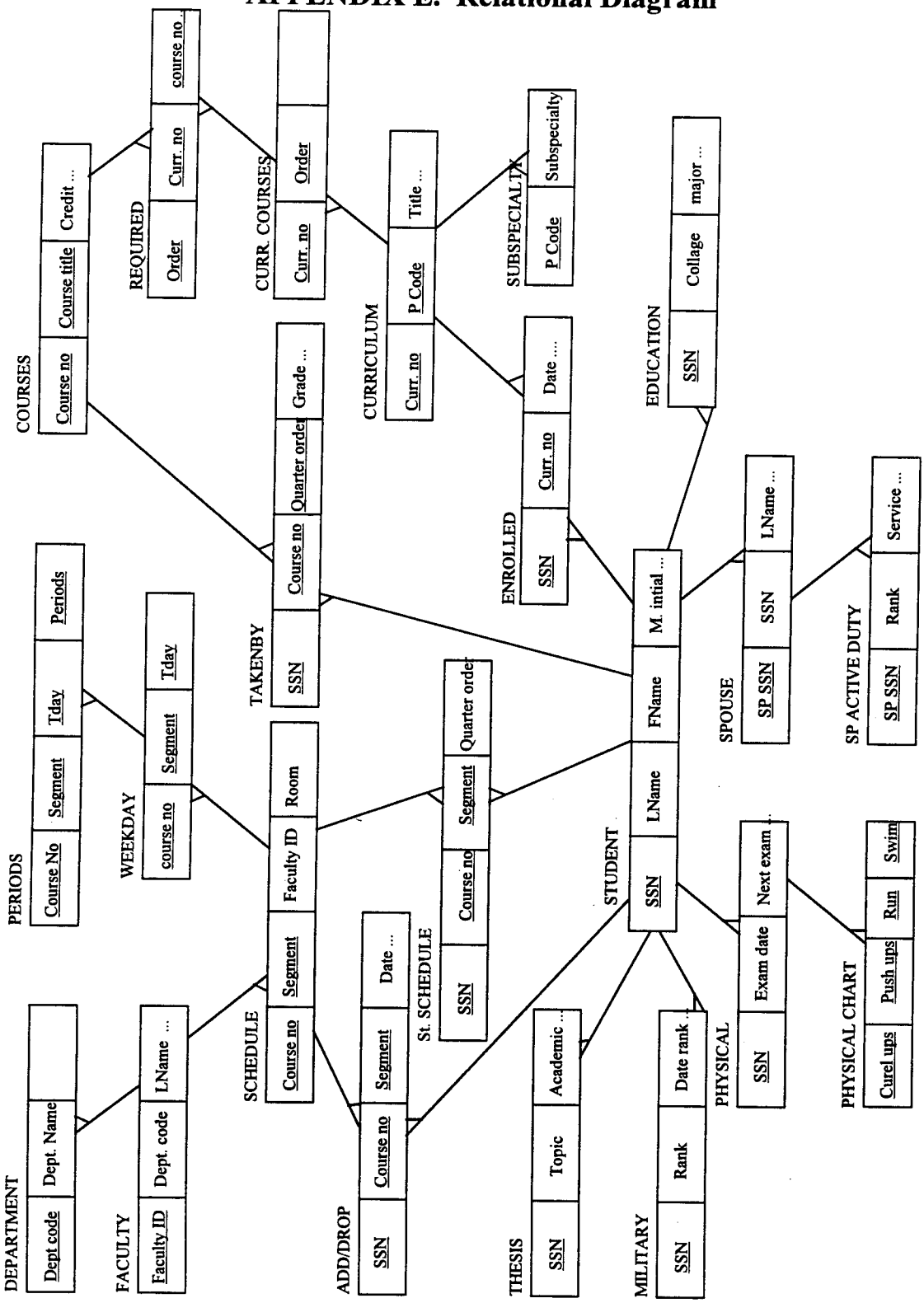
52. **Delete Courses Details (5.1.4P)**

a. Course number (Personnel/Admin Office).

b. Confirmation of deletion.

c. Confirm deletion actually will remove the record from Schedule data stores.

APPENDIX E. Relational Diagram



APPENDIX F. Menus, Forms, and Reports

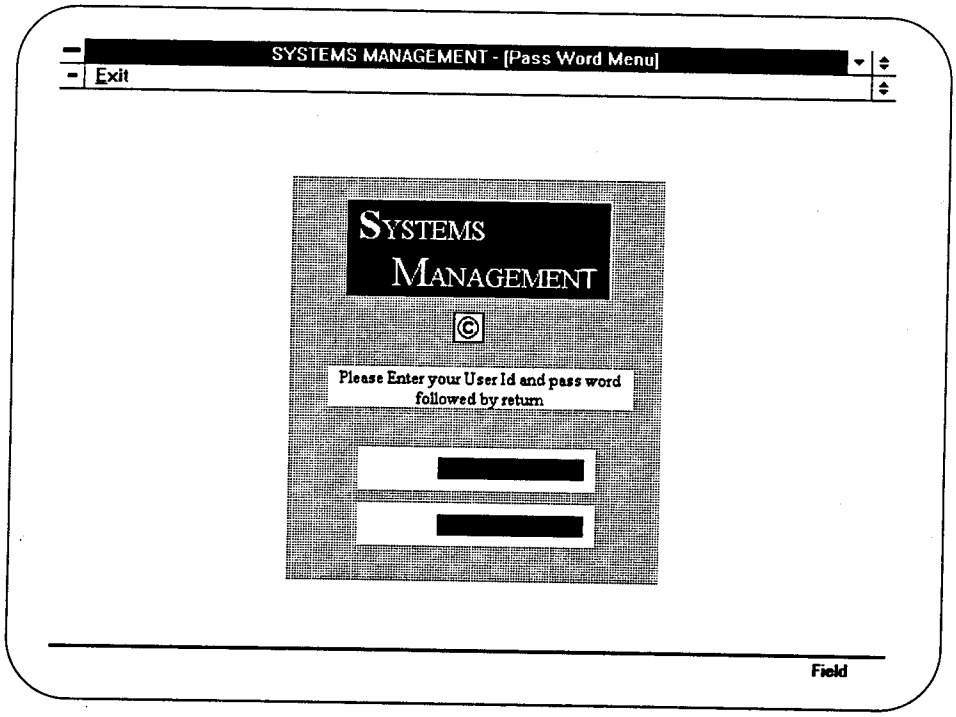


Figure [1] Login Screen

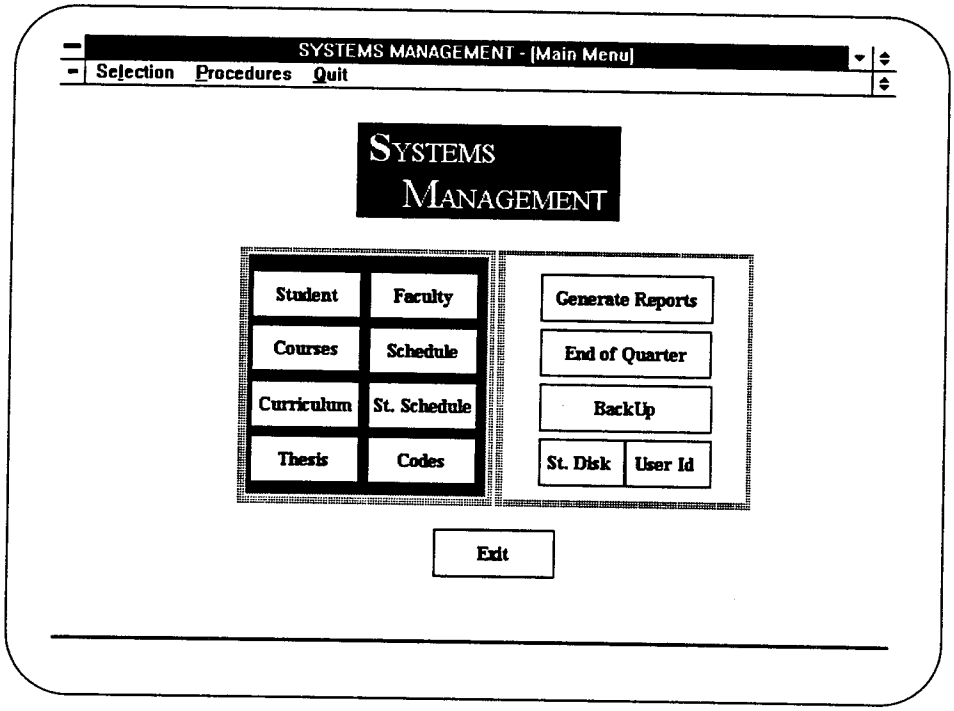


Figure [2] Main menu

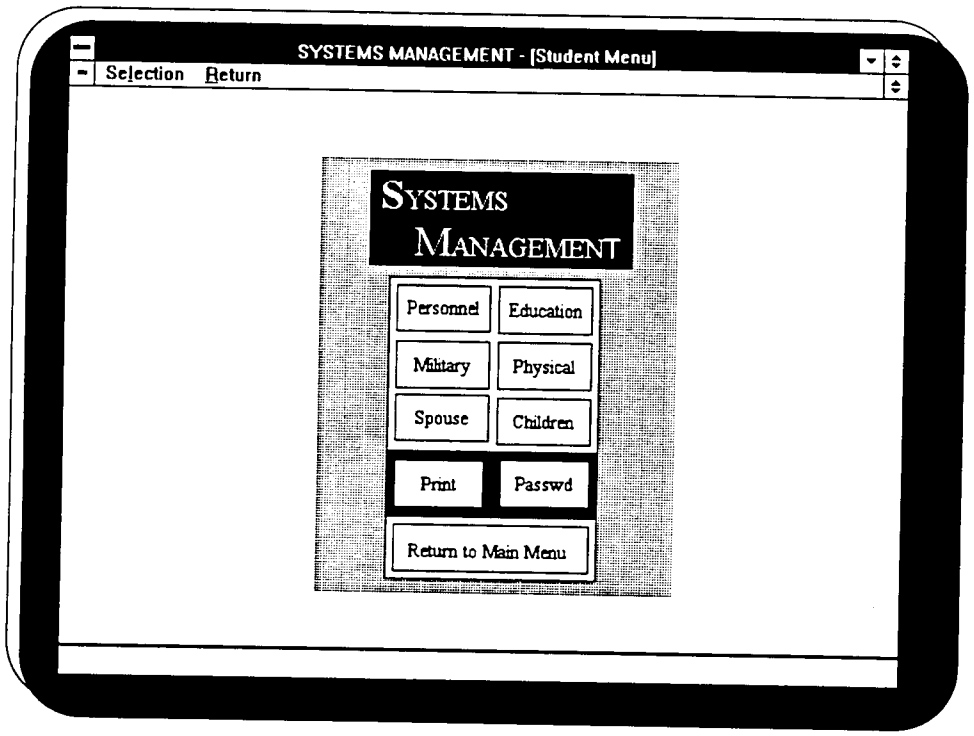


Figure [3] Student Submenu

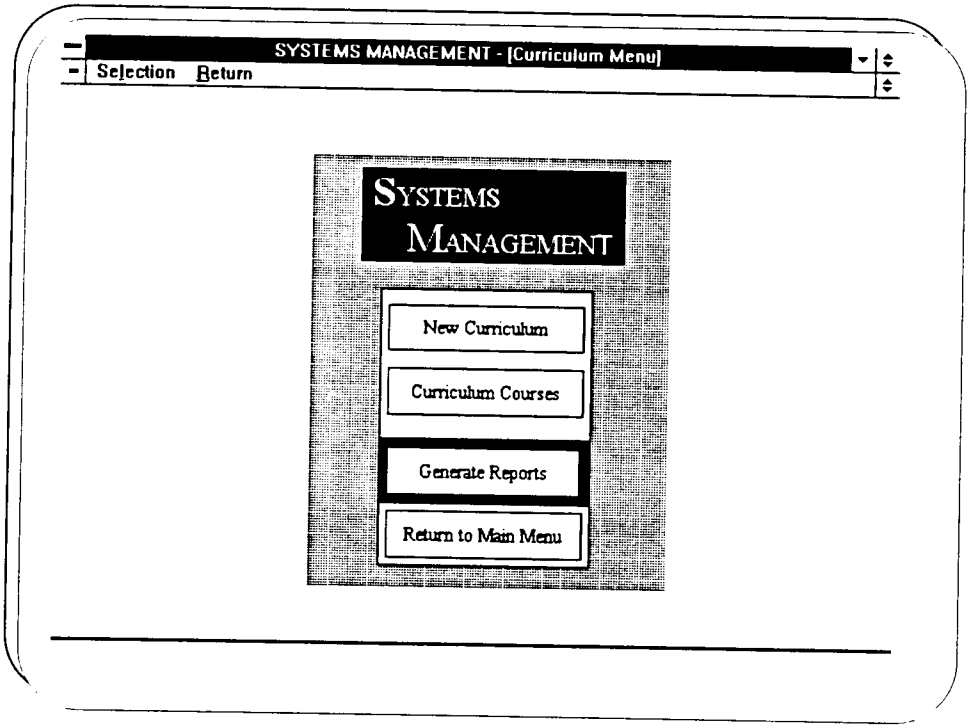


Figure [4] Curriculum submenu

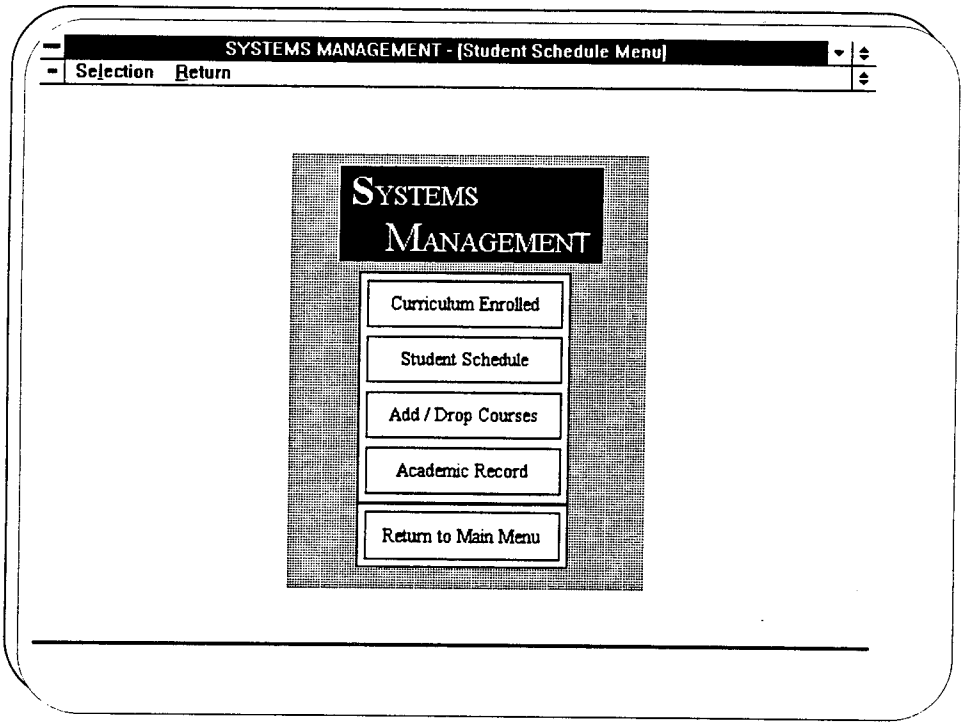


Figure [5] Student Schedule menu

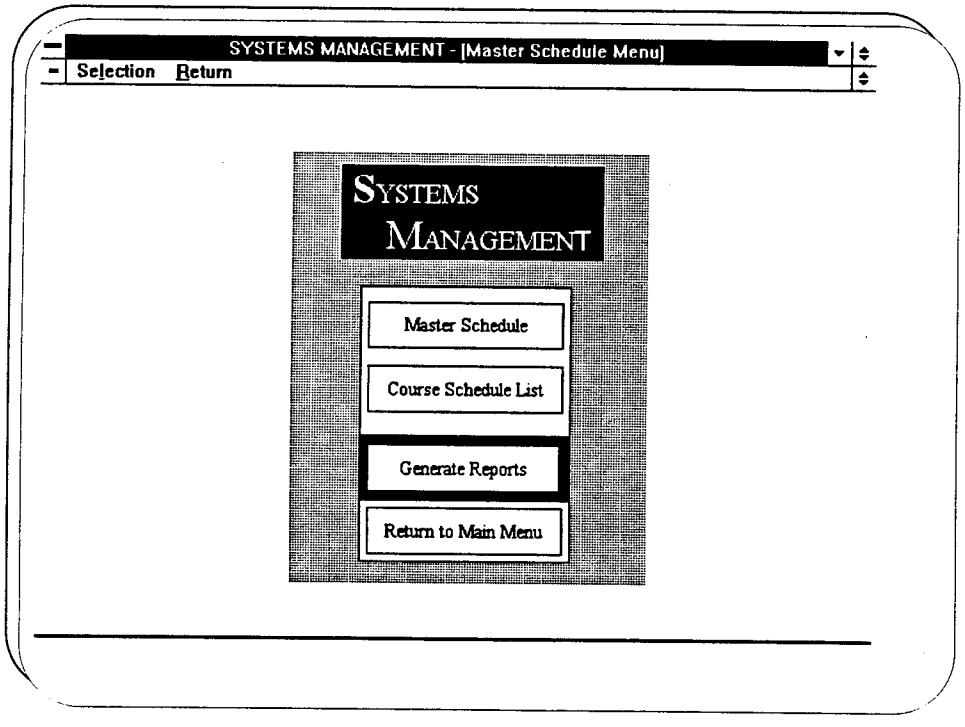


Figure [6] Master Schedule submenu

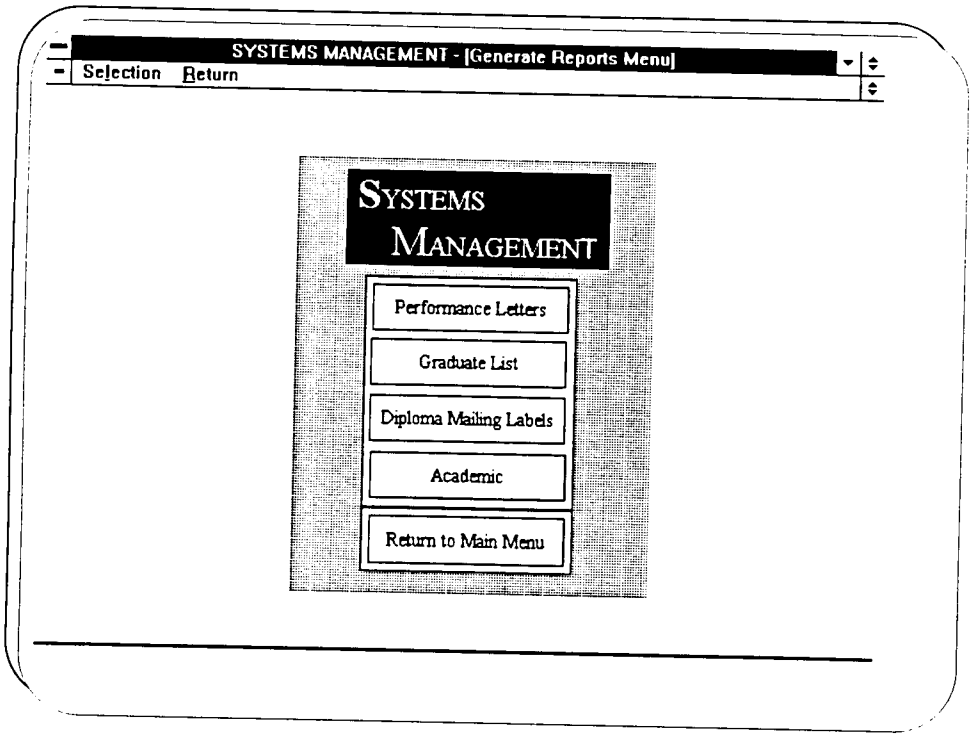


Figure [7] Generate Reports submenu

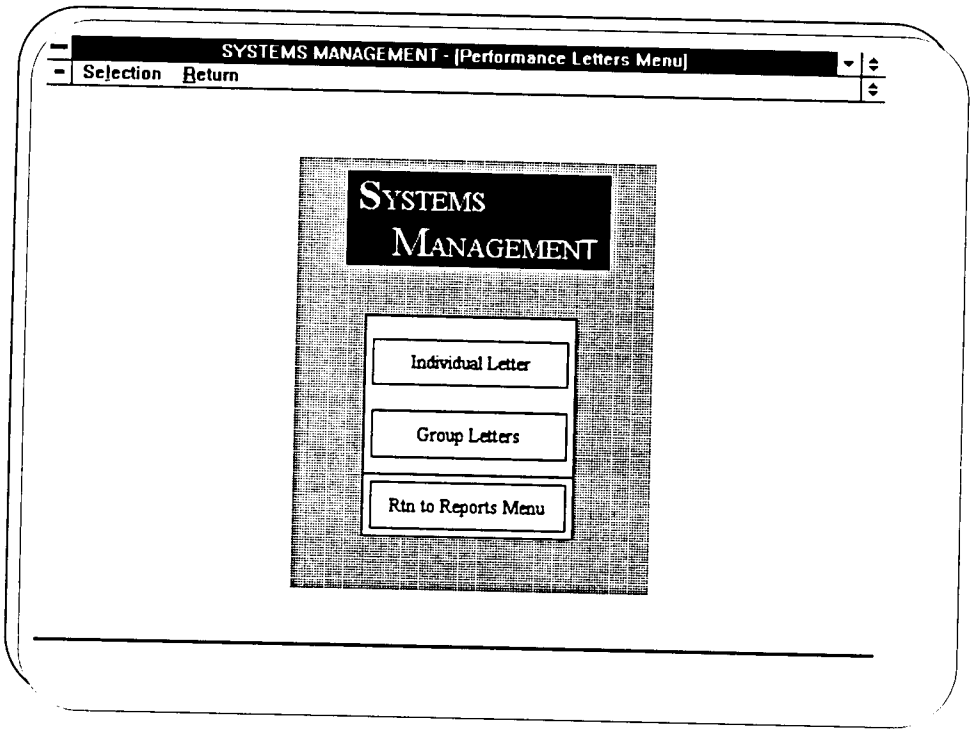


Figure [8] Performance Letters submenu

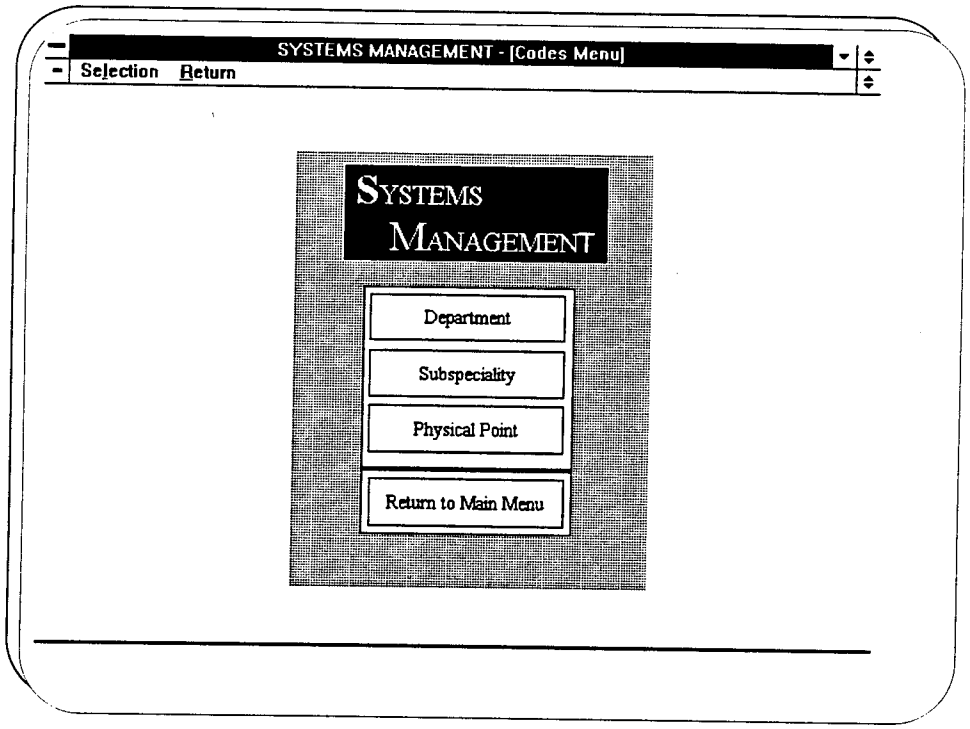


Figure [9] Codes submenu

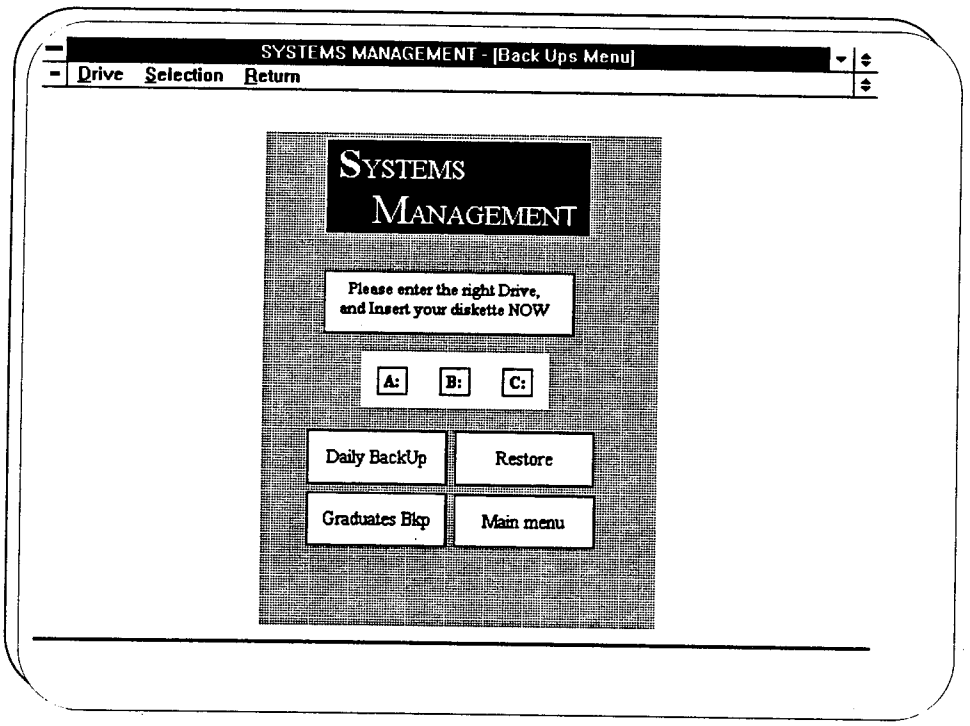


Figure [10] Back up submenu

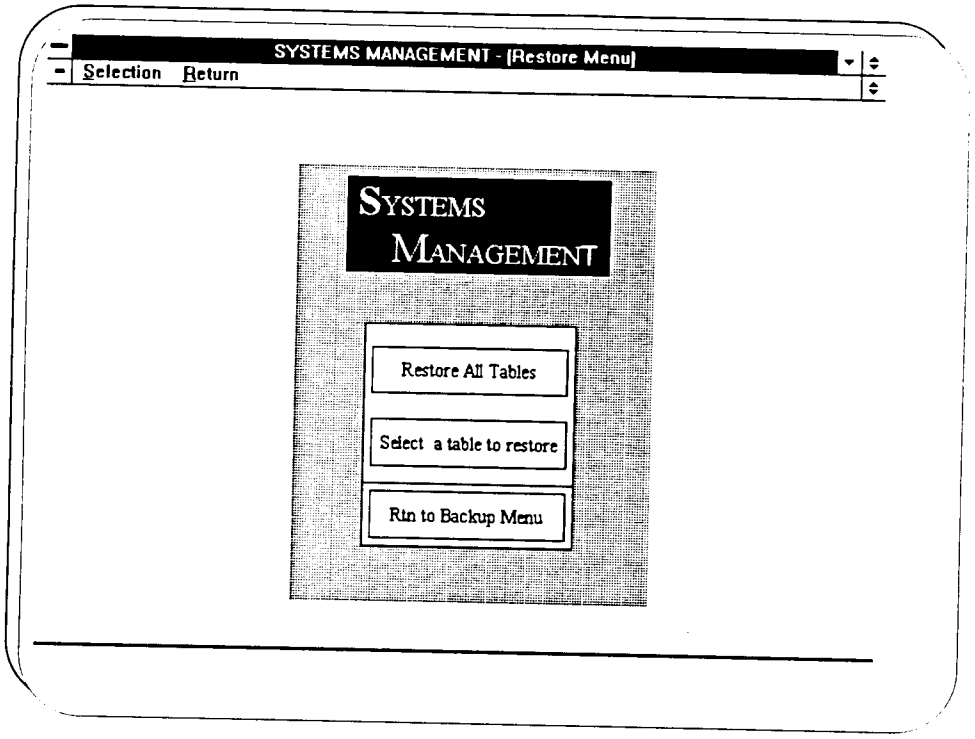


Figure [11] Restore submenu

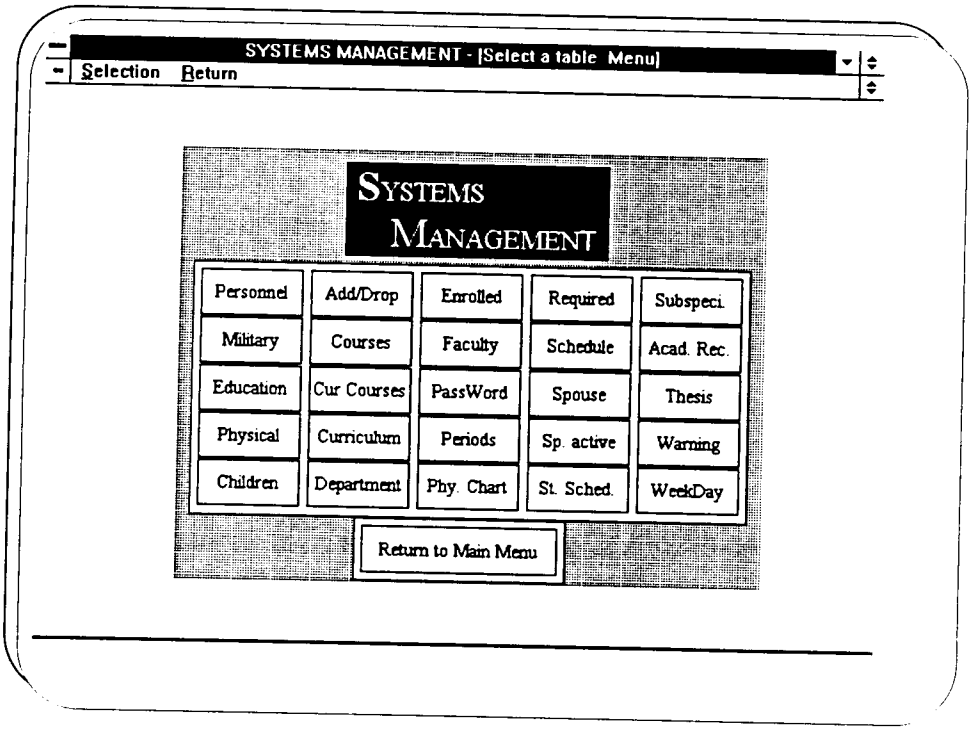


Figure [12] Select a Table to Restore submenu

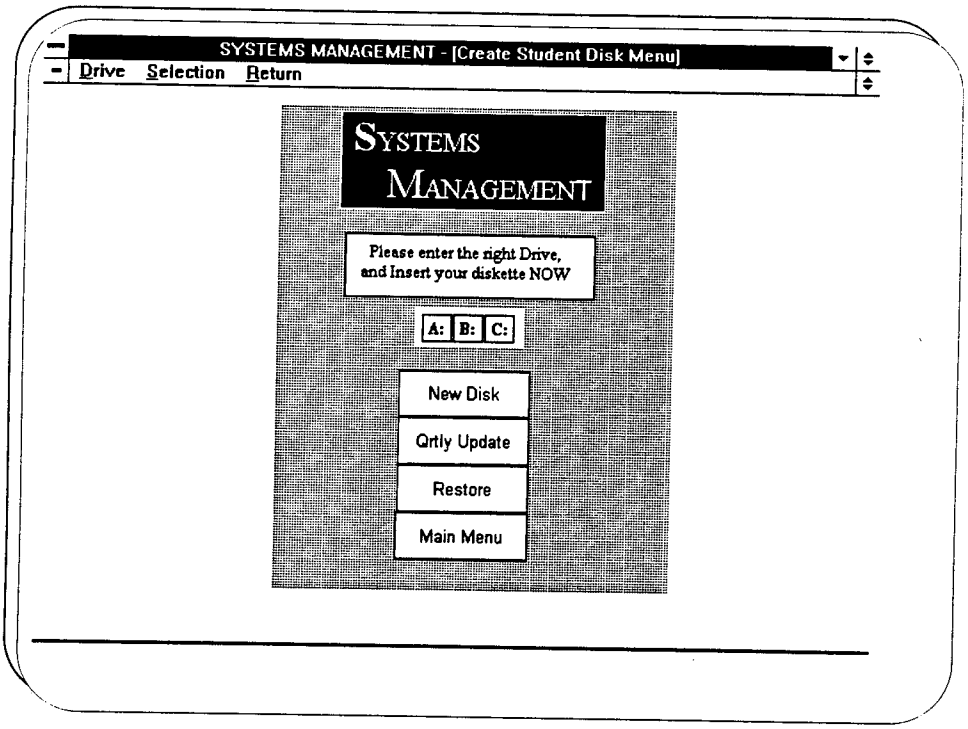


Figure [13] Student Disk submenu

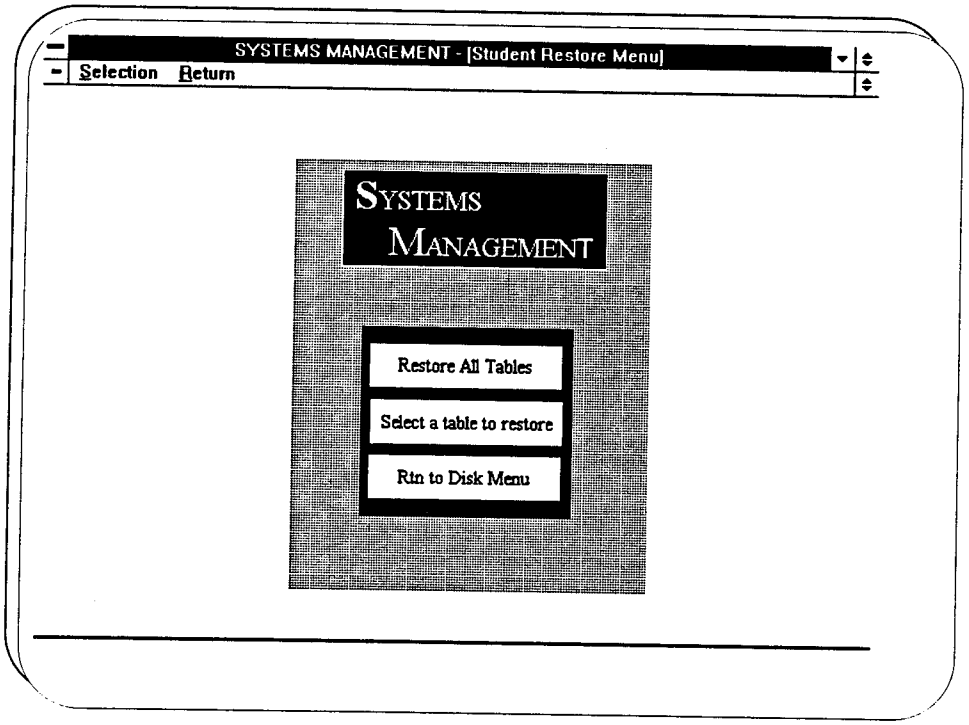


Figure [14] Restore (student disk) submenu

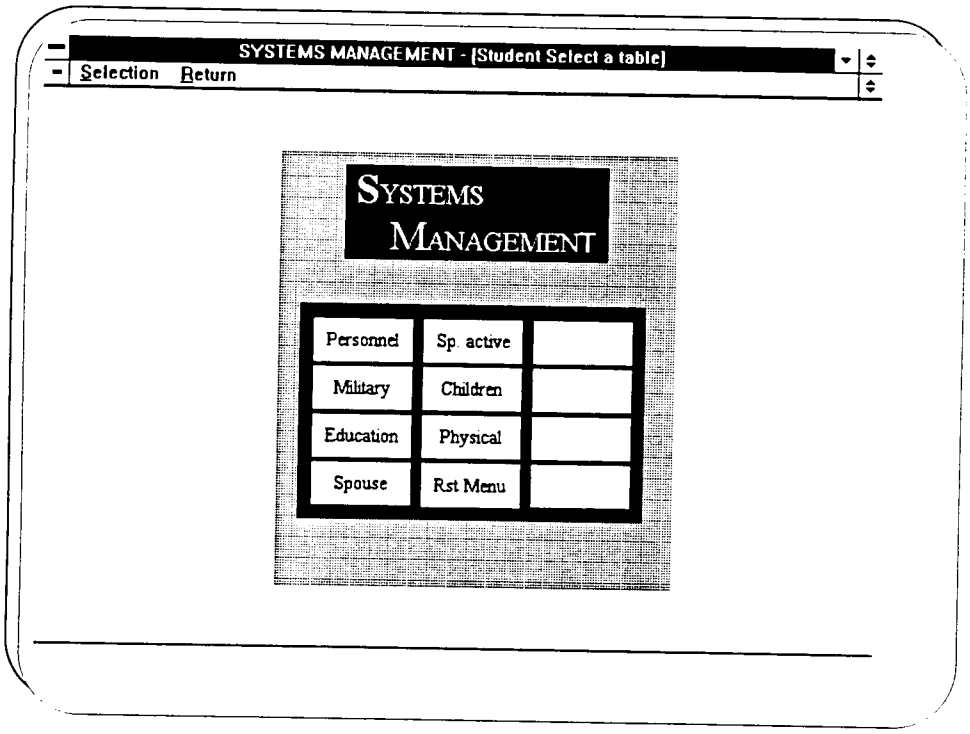


Figure [15] Select a Table to Restore (student disk)

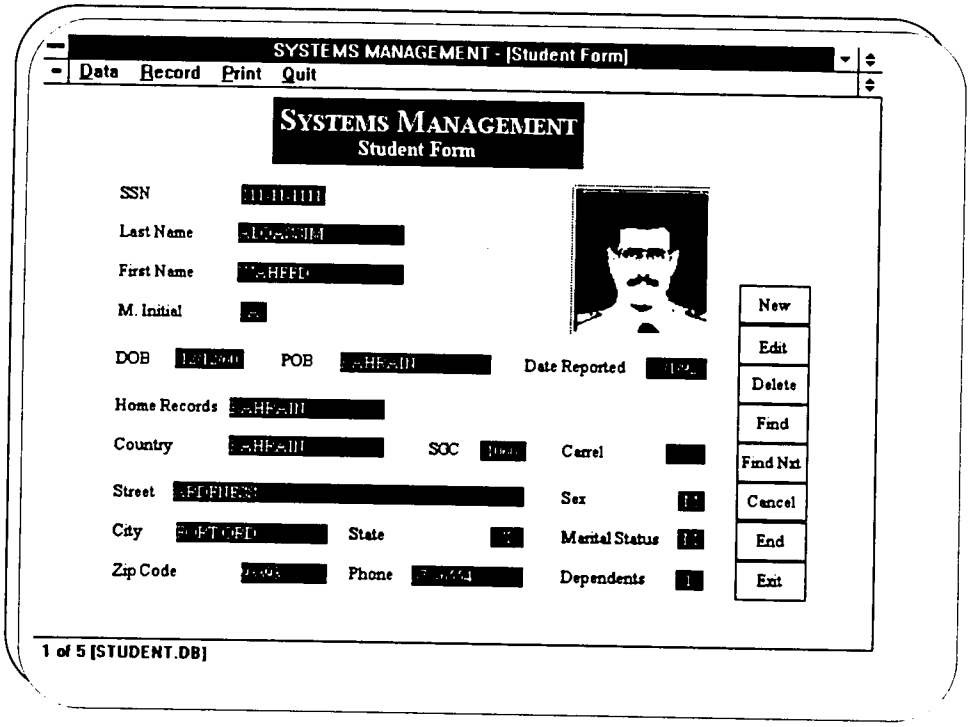


Figure [16] Student Form

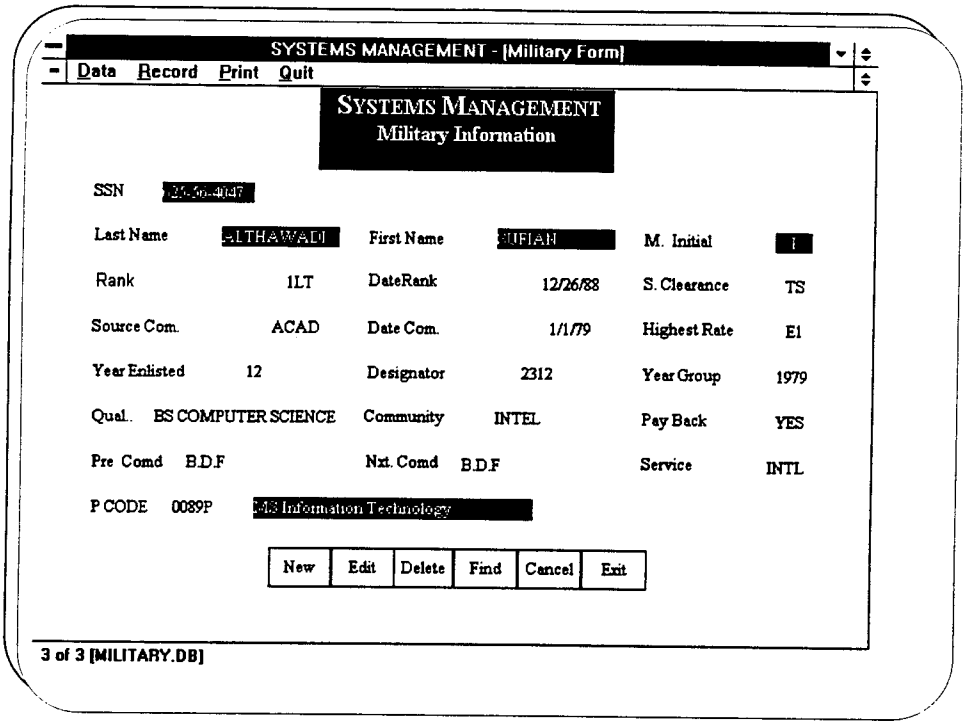


Figure [17] Military Form

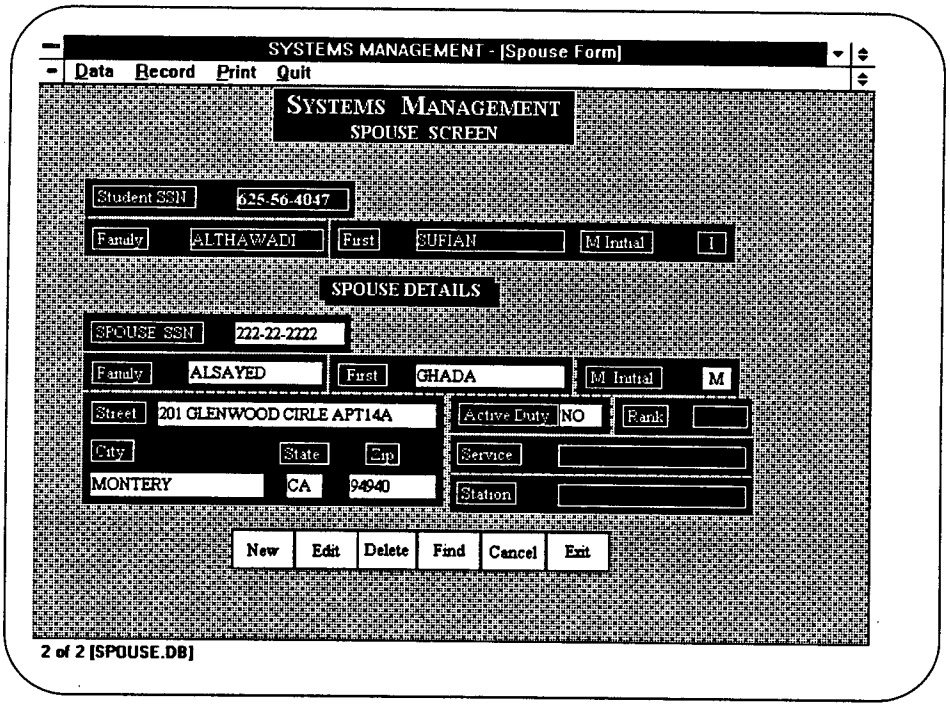


Figure [18] Spouse Form

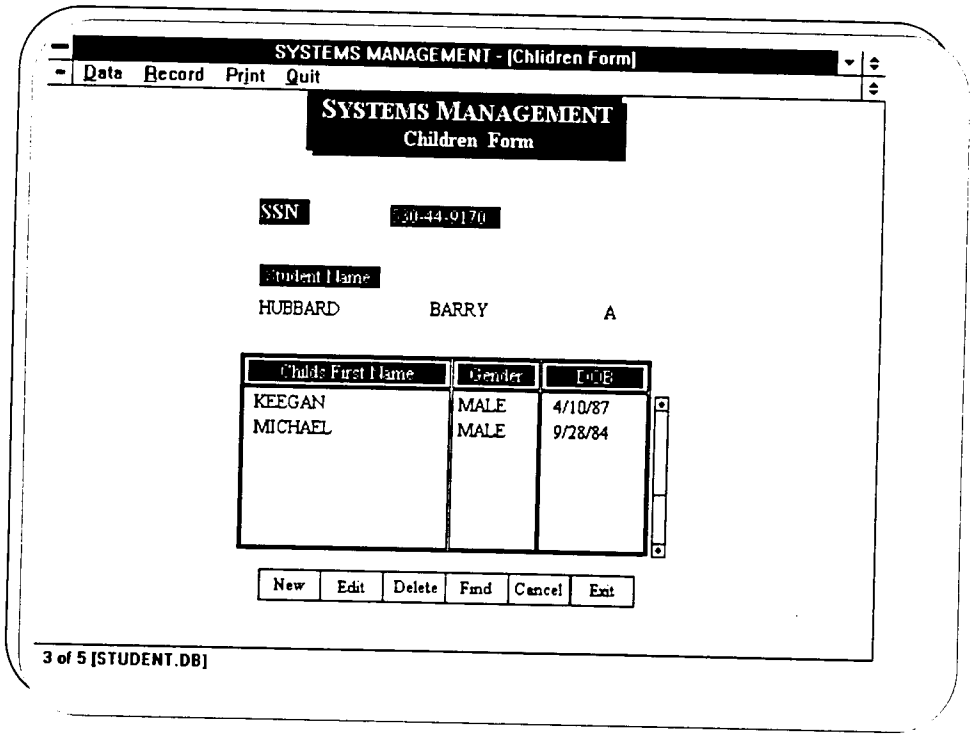


Figure [19] Children Form

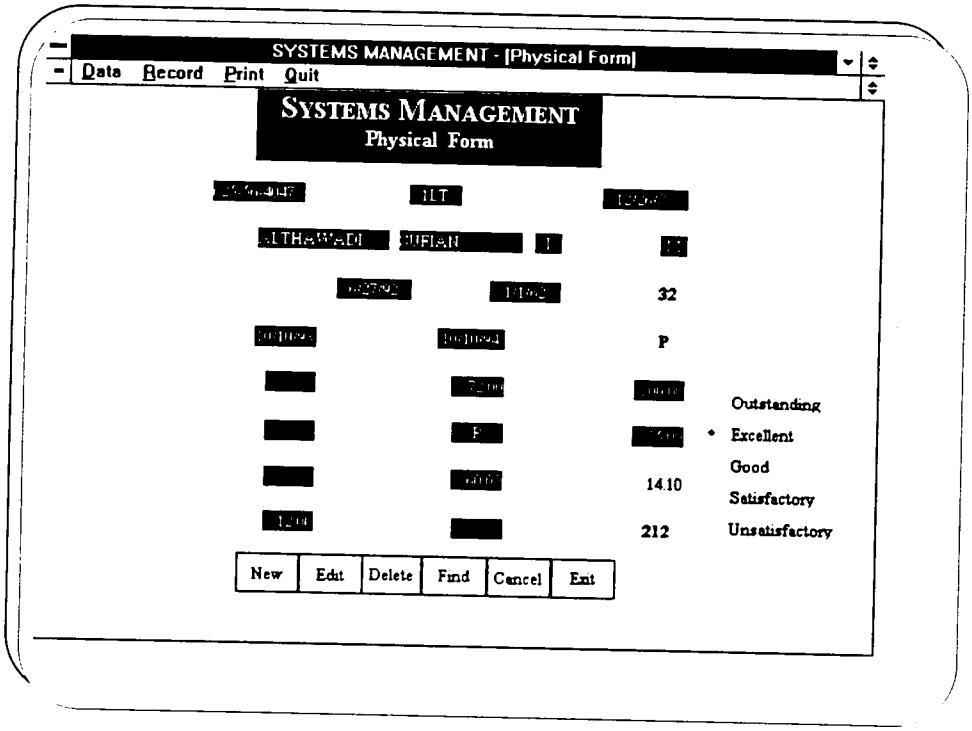


Figure [20] Physical Form

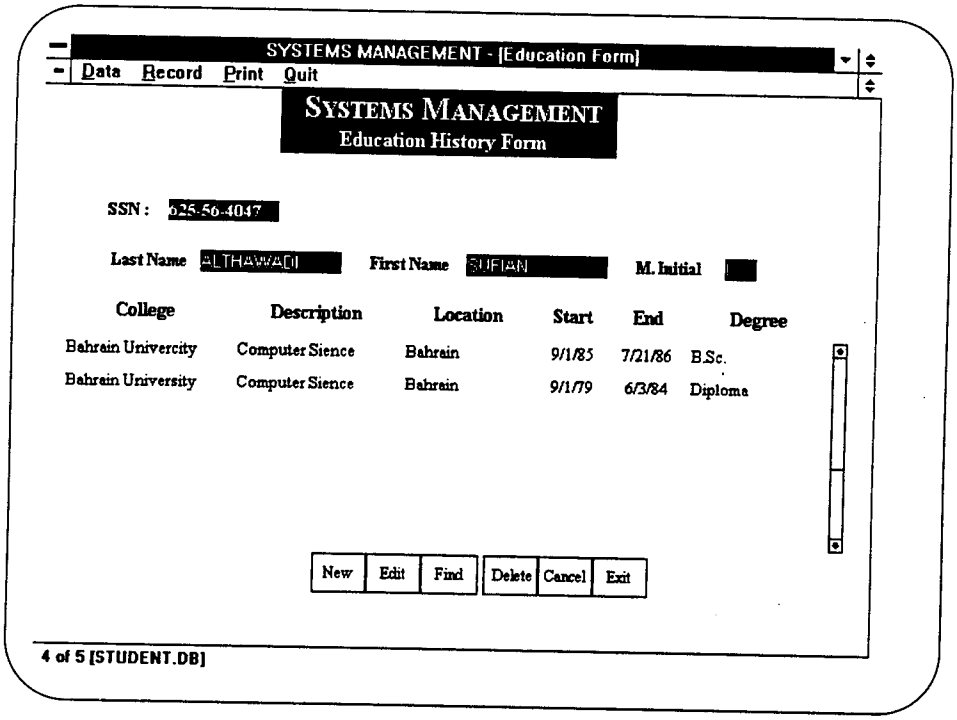


Figure [21] Education Form

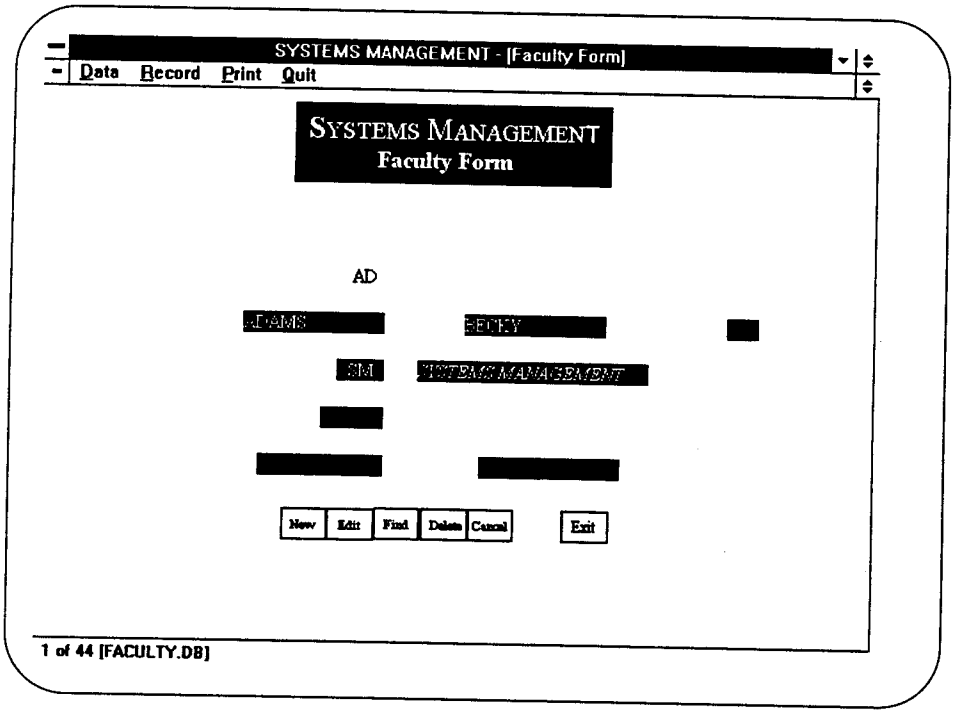


Figure [22] Faculty Form

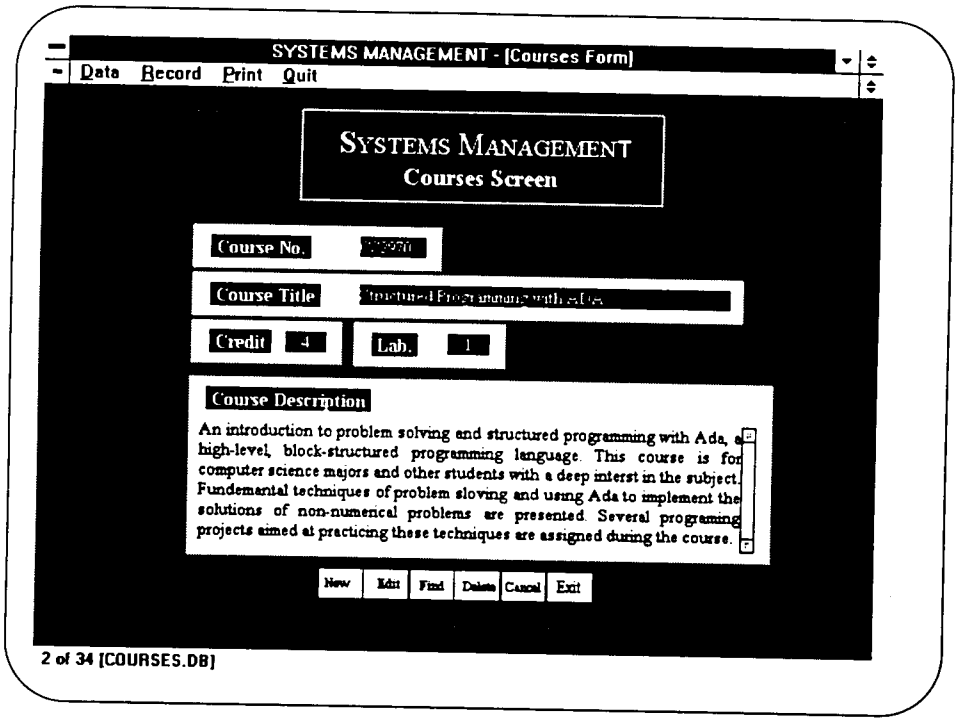


Figure [23] Courses Form

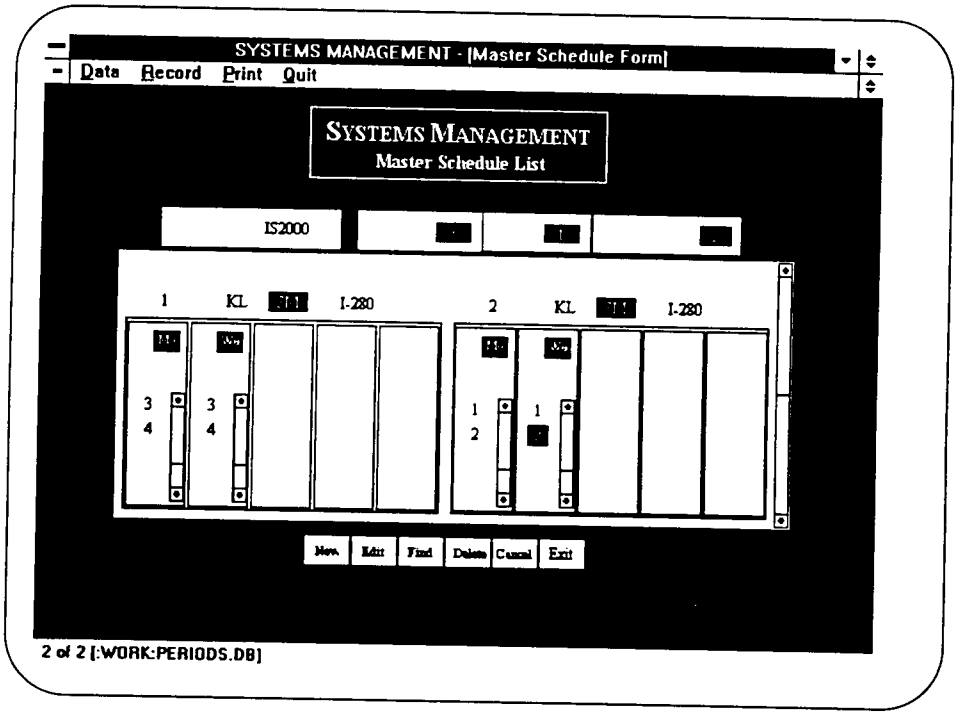


Figure [24] Master Schedule Form

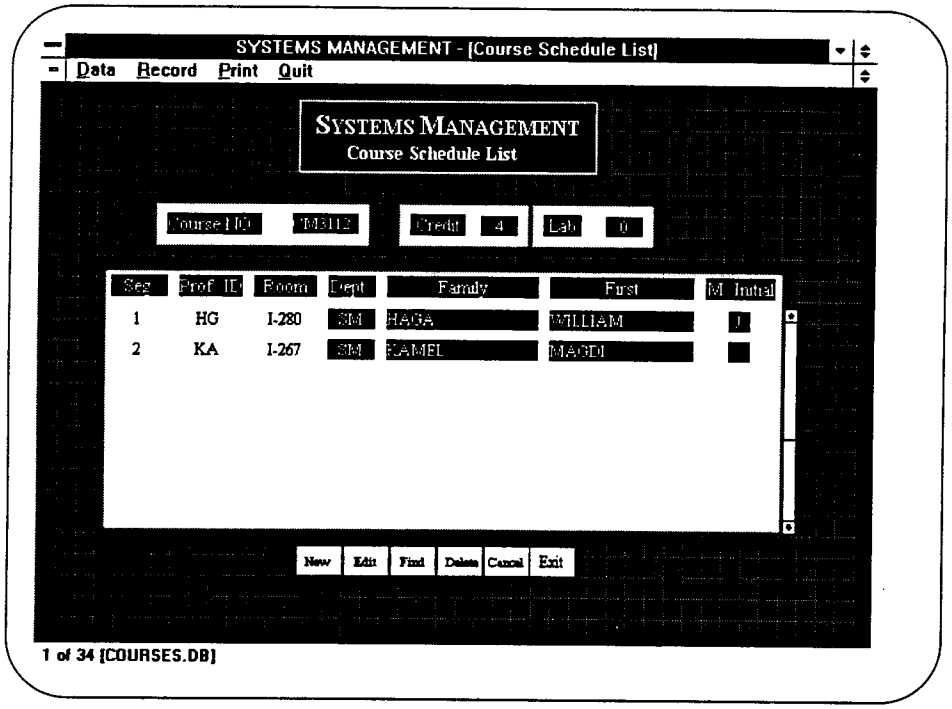


Figure [25] Course schedule Form

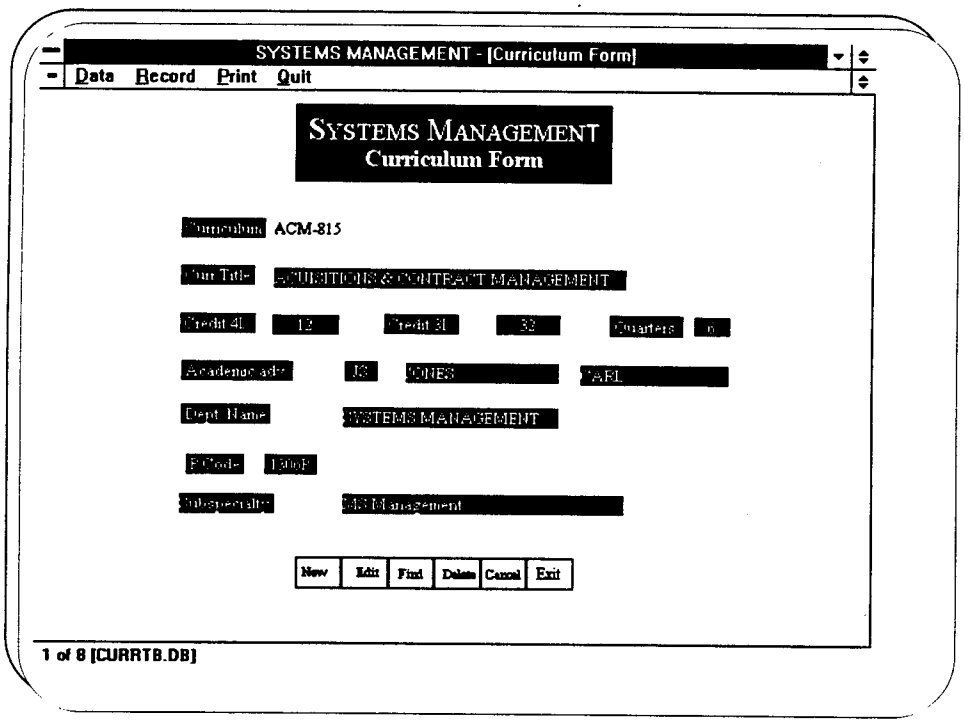


Figure [26] CurriculumForm

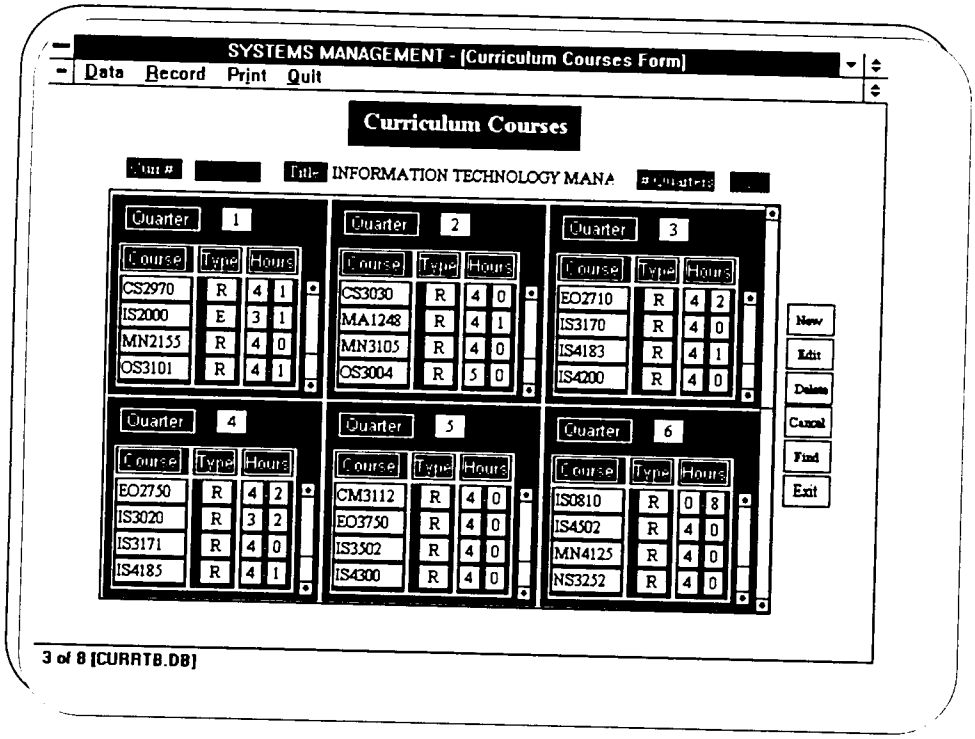


Figure [27] Curriculum Courses Form

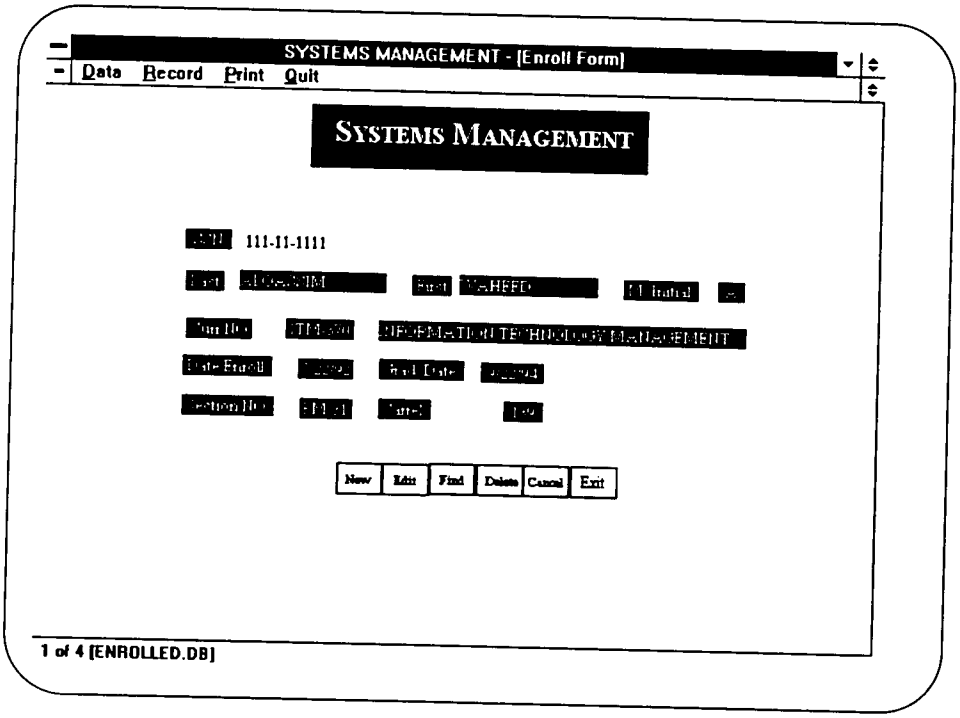


Figure [28] Enrolled Form

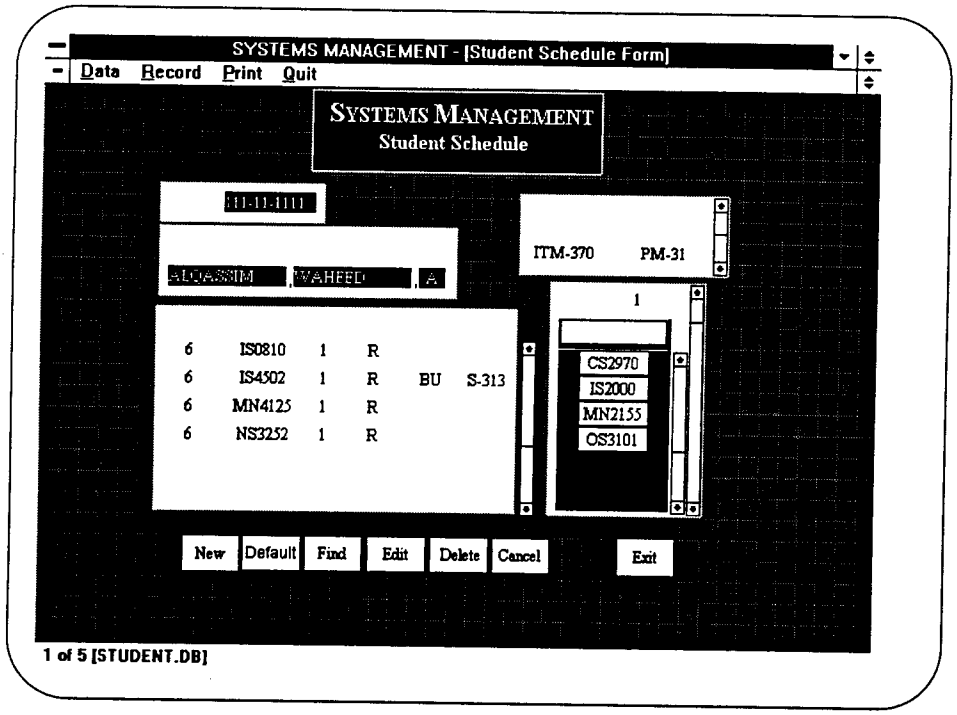


Figure [29] Student Schedule Form

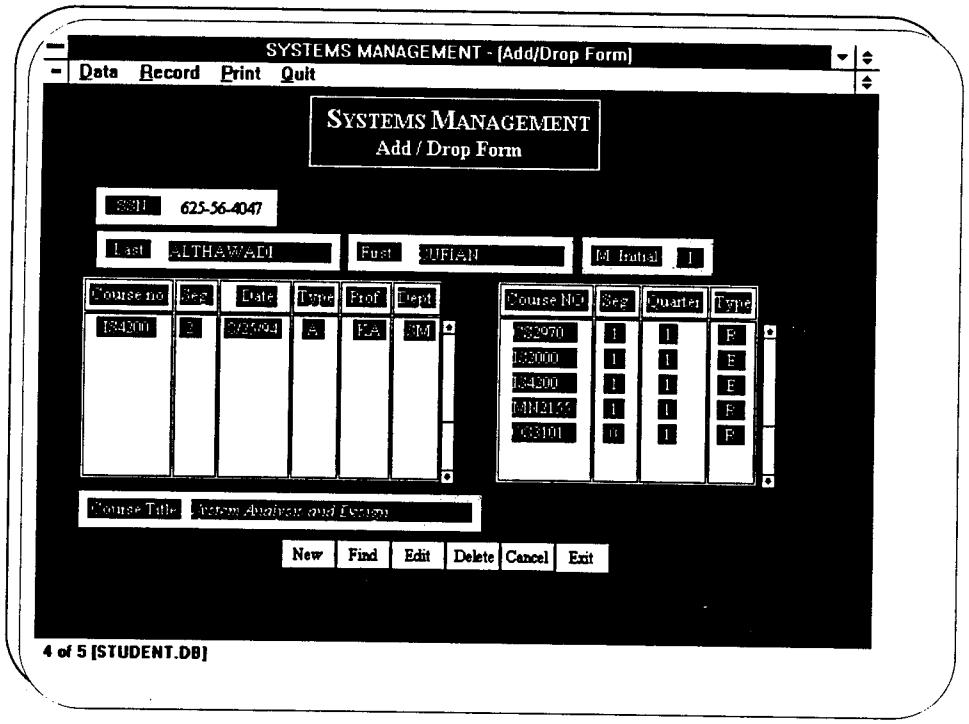


Figure [30] Add/Drop Form

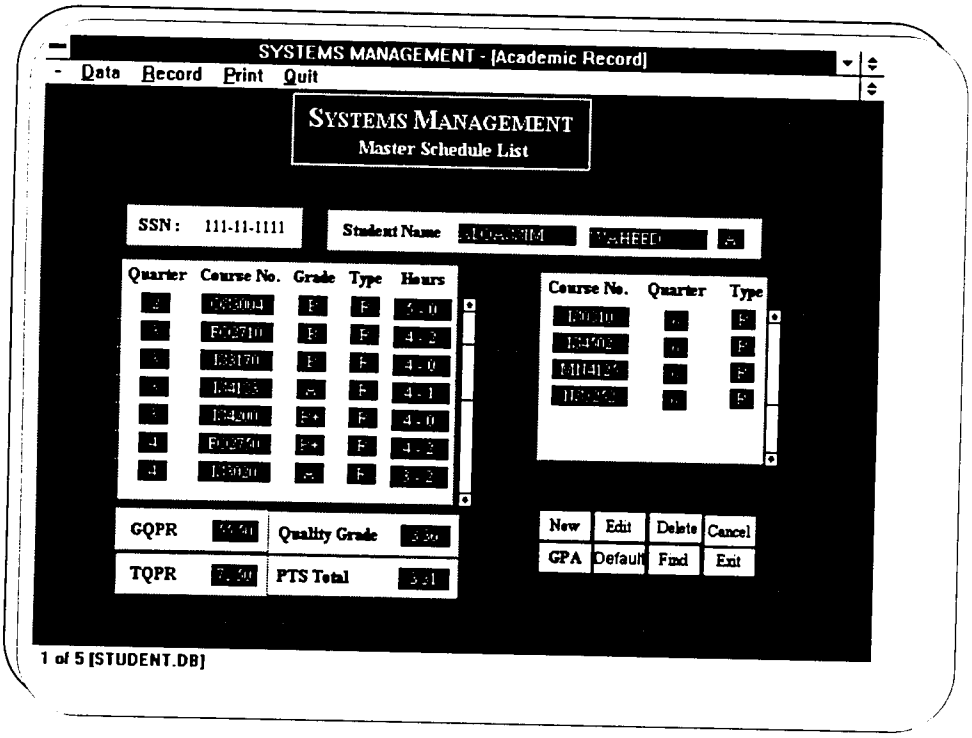


Figure [31] Academic record Form

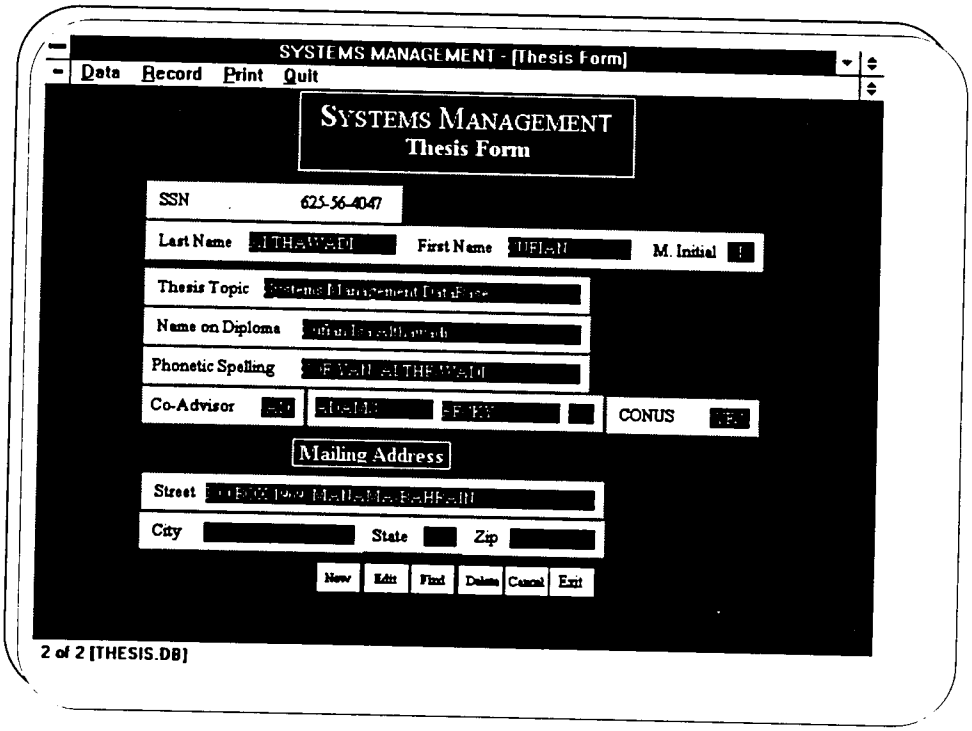


Figure [32] Thesis Form

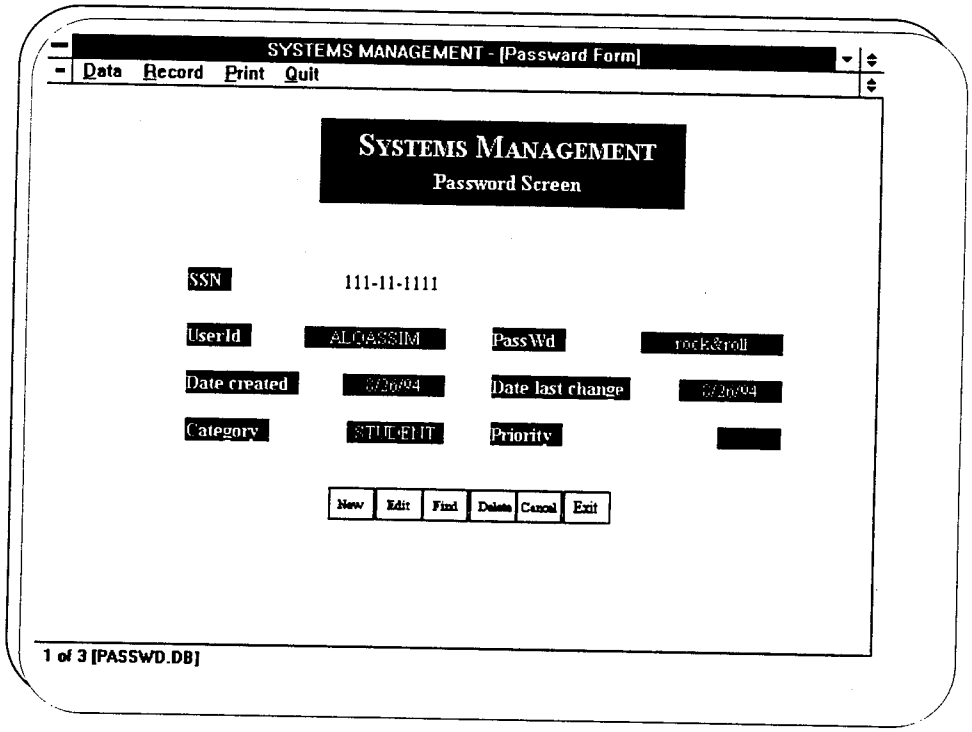


Figure [33] Password Form

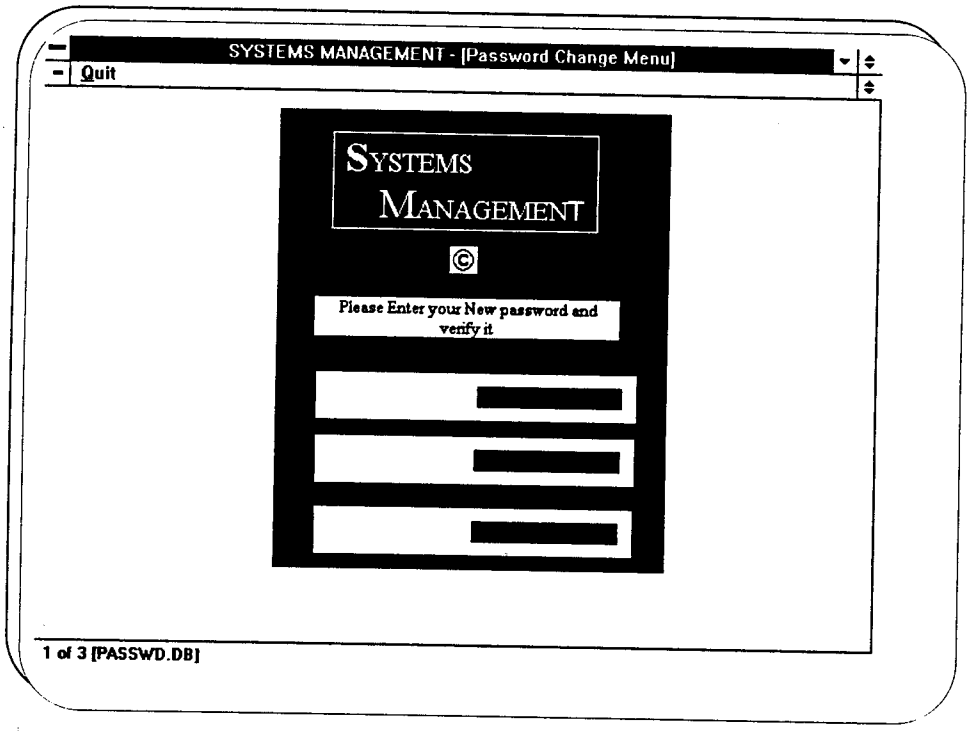


Figure [34] Password change Form

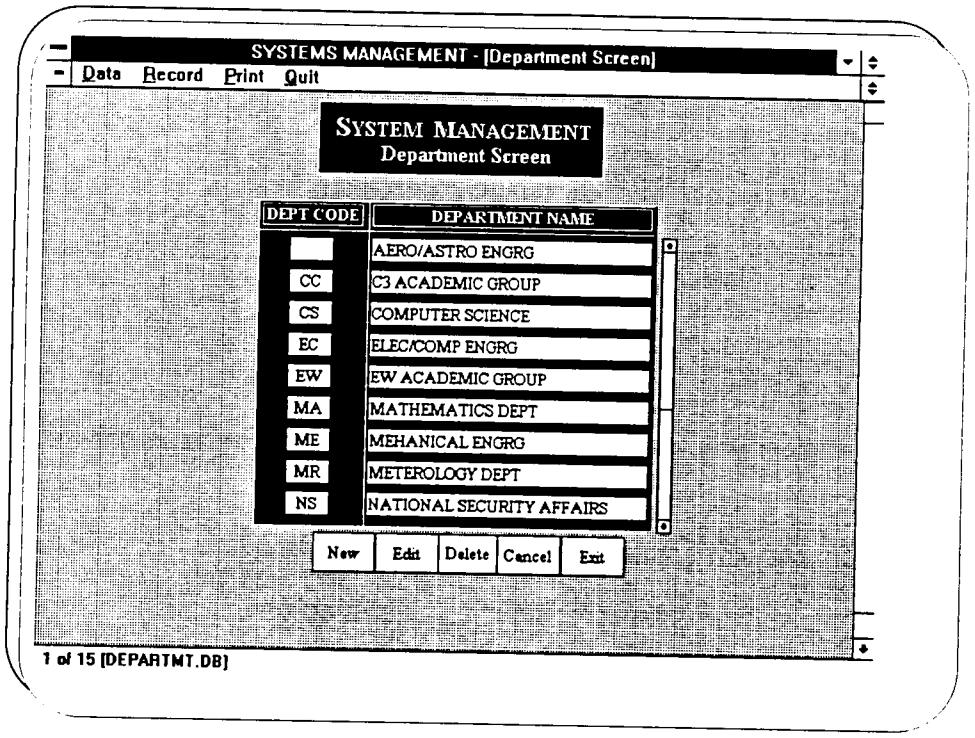


Figure [35] Department Form

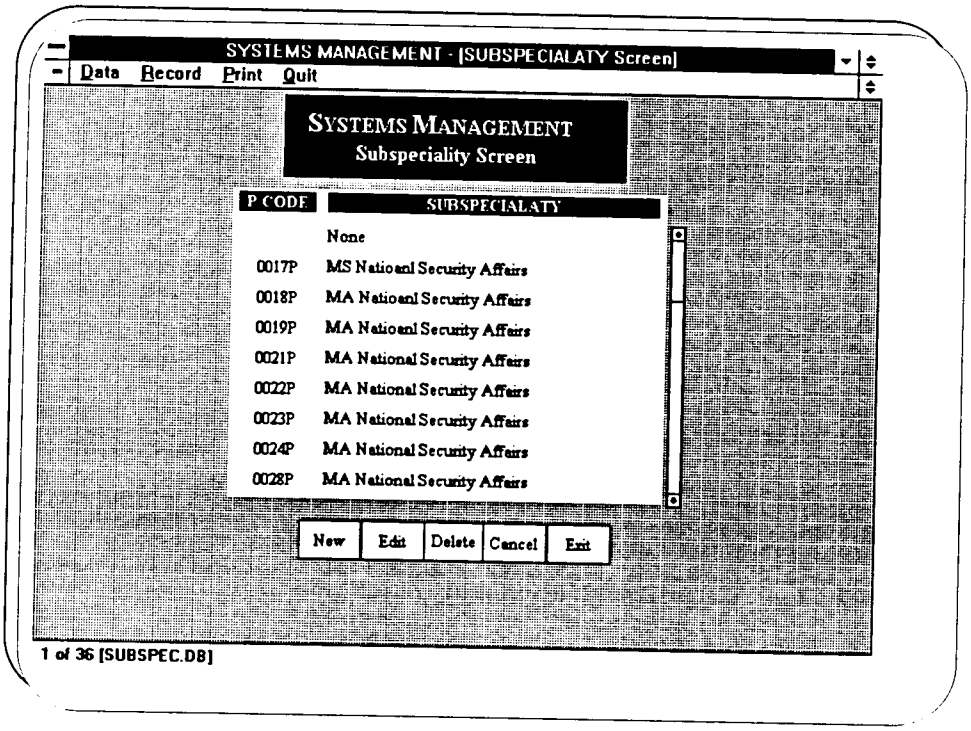


Figure [36] Subspecialty Form

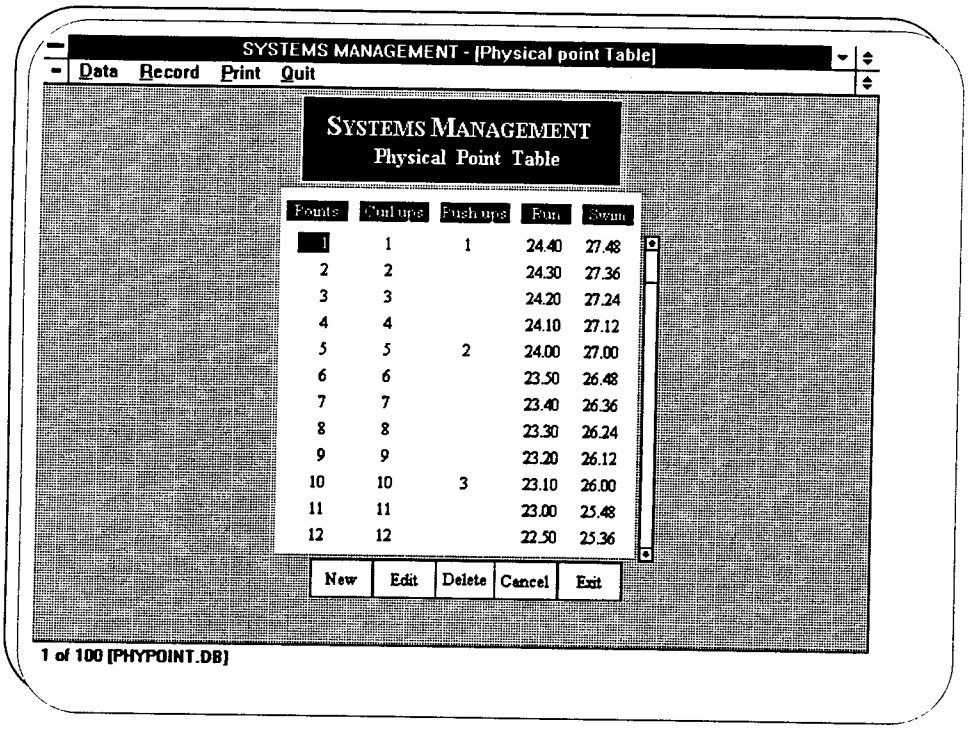


Figure [37] physical Chart Form

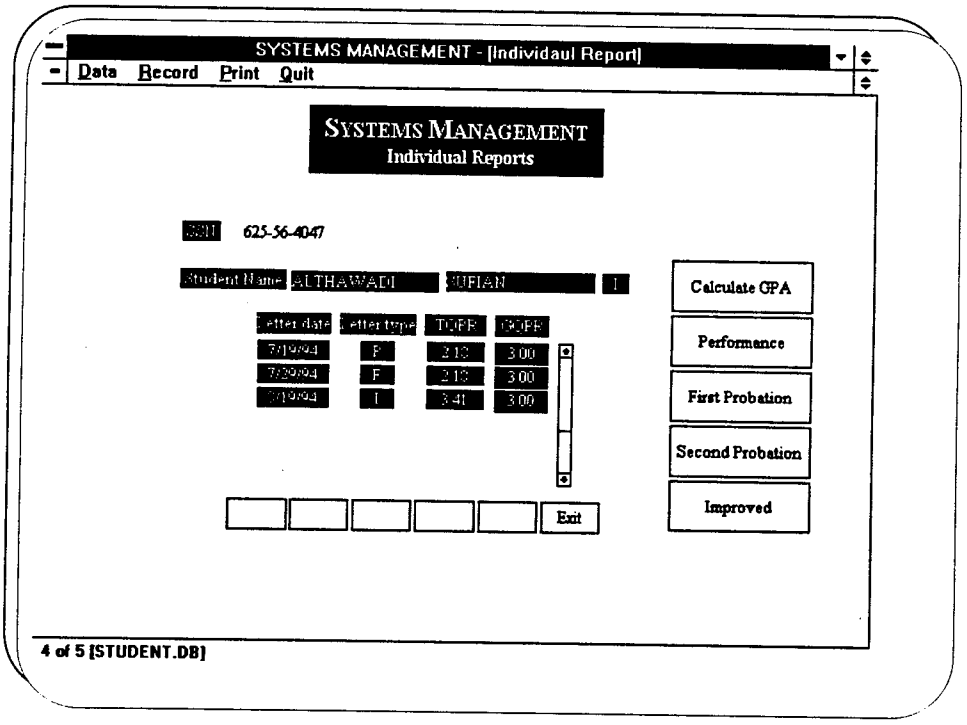


Figure [38] Individual report menu

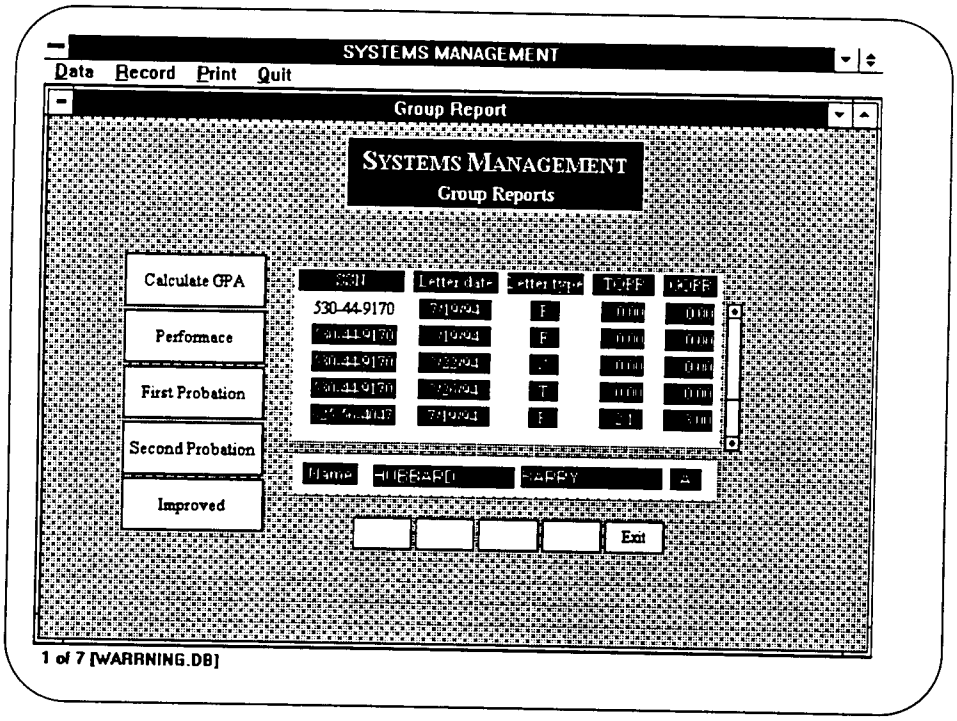


Figure [39] Group report menu

DEPARTMENT OF THE NAVY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5100

Tuesday, September 20, 1994

MEMORANDUM

From: Systems Management Curricular Officer/Academic Associate (ITM-370)

To: JOHNSON STEVE I , INTL , 625-56-4047

Subj: **ACADEMIC PERFORMANCE**

1. Areview of your academic transcript for the quarter ending SEPTEMBER /1994 reveals that your Graduate Quality Point Rating (GQPR) is 1.94 and your Total Quality Point Rating (TQPR) is 1.24 . The purpose of this memo is to remind you that a minimum GQPR of 3.00 and a TQPR of 2.75 must be obtained in order to receive a Master of Science in Management degree.

2. We trust that you are making every effort to bring up your grades. You are not being placed on academic probation at this time.

3. If you have any questions or need any assistance, please contact me or you Academic Associate.

M. P. Tryon
CDR, SC, USN

DEPARTMENT OF THE NAVY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5100

Tuesday, September 20, 1994

MEMORANDUM

From: Systems Management Curricular Officer/Academic Associate (ITM-370)

To: JOHNSON STEVE I , INTL , 625-56-4047

Subj: **NOTIFICATION OF ACADEMIC PERFORMANCE**

1. A review of your academic transcript for the quarter ending SEPTEMBER /1994 reveals that your Graduate Quality Point Rating (GQPR) is 1.94 and your Total Quality Point Rating (TQPR) is 1.24 . The purpose of this memo is to advise you that a minimum GQPR of 3.00 and a TQPR of 2.75 must be obtained in order to receive a Master of Science in Management degree.

2. In view of the foregoing, you are notified that you have been placed on academic probation. Failure to meet the minimum standards, depending on subsequent performance, may result in disenrollment.

3. Extenuating circumstances, or a need for additional assistance or instruction in assigned courses, should be discussed with the curricular officer or your Academic Associate.

M. P. Tryon
CDR, SC, USN

DEPARTMENT OF THE NAVY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5100

Tuesday, September 20, 1994

MEMORANDUM

From: Systems Management Curricular Officer/Academic Associate (ITM-370)

To: JOHNSON STEVE I , INTL , 625-56-4047

Subj: **NOTIFICATION OF ACADEMIC PROBATION**

1. A review of your academic transcript for the quarter ending SEPTEMBER /1994 reveals that your Graduate Quality Point Rating (GQPR) is 1.94 and your Total Quality Point Rating (TQPR) is 1.24 . The purpose of this memo is to advise you that a minimum GQPR of 3.00 and a TQPR of 2.75 must be obtained in order to receive a Master of Science in Management degree.

2. In view of the foregoing, you will remain on academic probation. This is your third quarter on academic Probation, you must earn A's in all your courses in order to meet graduation requirements.

3. Extenuating circumstances, or a need for additional assistance or instruction in assigned courses, should be discussed with the curricular officer or your Academic Associate.

M. P. Tryon
CDR, SC, USN

DEPARTMENT OF THE NAVY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5100

Tuesday, September 20, 1994

MEMORANDUM

From: Systems Management Curricular Officer/Academic Associate (ITM-370)

To: JOHNSON STEVE I , INTL , 625-56-4047

Subj: **IMPROVED ACADEMIC PERFORMANCE**

1. A review of your academic transcript for the quarter ending SEPTEMBER /1994 reveals that your Graduate Quality Point Rating (GQPR) is 3.53 and your Total Quality Point Rating (TQPR) is 2.59.

2. The purpose of this memo is congratulate you on your improved grade point average. You have put fourth significant efforts towards achieving the academic standards required for a degree at the Naval Postgraduate School. You are to be commended for these efforts.

M. P. Tryon
CDR, SC, USN

GRADUATES

NAME : ALQASSIM , WAHEED , A
Dip name : Waheed Abdula Alqassim RANK : LT
Phonetic : Wa-heed Al-Qassim SERVICE : INTL
Advisor : AD
PAY BACK : NO NEXT COMMAND :
Dip street : P.O. BOX 774 MANAMA - BAHRAIN
Dip city : ,

NAME : ALBUSMAIT , KHALID , Y
Dip name : Khalid Y. Albusmait RANK :
Phonetic : KHA-LED AL-BUSS-MAIT SERVICE :
Advisor : HA
PAY BACK : NEXT COMMAND :
Dip street : 253 AVENUE 1, MUHARRAQ
Dip city : BAHRAIN ,

NAME : HUBBARD , BARRY , A
Dip name : Barry Hubbard RANK : LTCDR
Phonetic : BARRY HUBBARD SERVICE : USN
Advisor : HA
PAY BACK : NO NEXT COMMAND :
Dip street : 123 CUSTOM ST
Dip city : PEBBEL BEACH , CA 93942-

NAME : JOHNSON STEVE , I
Dip name : Sufian Isa Althawadi RANK : 1LT
Phonetic : SOF-YAN ALTHE-WADI SERVICE : INTL
Advisor : AD
PAY BACK : YES NEXT COMMAND : B.D.F
Dip street : 201 GLENWOOD CIRCLE #14 A
Dip city : MONTEREY , IN 34221-

NAME : ALTHAWADI , ISA , S
Dip name : Isa Sufian Althawadi RANK :
Phonetic : ESSA-SOF-YAN-AL-THE-WADI SERVICE :
Advisor : AD
PAY BACK : NEXT COMMAND :
Dip street : 5200 COE AVENUE #2151
Dip city : FORT ORD , CA 93941-

Courses Schedule for September, 1994

Curriculum No: ITM-370 / PM-31

Student Name	Courses				
ALQASSIM, WAHEED, A	IS0810	IS4502	MN4125	NS3252	
HUBBARD, BARRY, A	CS2970	IS2000	MN2155	OS3101	
ALTHAWADI, SUFIAN, I	CS2970	IS2000	IS4200	MN2155	OS3101

Student list by Country September, 1994

Country : BAHRAIN **# of Student :** 3

ALQASSIM, WAHEED, A

ALBUSMAIT, KHALID Y

ALTHAWADI, SUFIAN, I

Country : USA **# of Student :** 2

HUBBARD, BARRY, A

ALTHAWADI, ISA, S

ALQASSIM , WAHEED , A

P.O. BOX 774 MANAMA - BAHRAIN
,

HUBBARD , BARRY ,

123 CUSTOM ST
PEBBEL BEACH, CA 93942-

ALTHAWADI , ISA , S

5200 COE AVENUE #2151
FORT ORD, CA 93941-

ALBUSMAIT , KHALID , Y

253 AVENUE 1, MUHARRAQ
BAHRAIN,

ALTHAWADI , SUFIAN , I

201 GLENWOOD CIRCLE #14 A
MONTEREY IN 34221-

APPENDIX G. Logic for Menus and Submenus

Object : passForm

MethodName : Var

Source :
```
Var
        formName        Form
        userCategory,userSSN String
        newDrive,copyName               String
endVar
```

Object : passForm

MethodName : close

Source :
```
method close(var eventInfo Event)
if eventInfo.isPreFilter()
        then
                ; This code executes for each object on the form.

        else
                ; This code executes only for the form.
                showSpeedBar()
                removeMenu()
endif
endmethod
```

Object : passForm

MethodName : arrive

Source :
```
method arrive(var eventInfo MoveEvent)
Var
        thisApp Application
endVar
if eventInfo.isPreFilter()
        then
                ; This code executes for each object on the form.

        else
                ; This code executes only for the form.
                thisApp.setTitle("SYSTEMS MANAGEMENT")
                if Not isMaximized() then
                        maximize()
                endIf
                hideSpeedBar()
                newDrive=""
                disableDefault
endif
endmethod
```

Object : stselectPage

MethodName : arrive

Source :
```
method arrive(var eventInfo MoveEvent)
if Not isMaximized() then
        maximize()
endIf
```

```
setTitle("Student Select a table")
endmethod
```

Object : stselectPage

MethodName : copyTB

Source :

```
method copyTB()
Var
        tmpTb Table
endVar
if msgQuestion("Please Confirm ! ...",
                "Do you want to restore the < "+copyName+" > table form the Student
copy?") = "Yes" then
        tmpTb.attach(newDrive+copyName)
        tmpTb.add(copyName,True,True)
        message("copy file form <"+newDrive+copyName+"> to <"+copyName+">")
endif
endmethod
```

Object : stselectPage.#Box265.#Box266.CoursesBttn1

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        strestorePage.moveTo()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn14

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
copyName="ST_SCHED.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn4

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
copyName="CHLIDREN.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn22

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
copyName="THESIS.db"
copyTB()
endmethod
```

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="SPACTIVE.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn12

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="SPOUSE.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn19

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="ENROLLED.db"
copyTB()

endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn9

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="ADDDROP.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn3

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="PHYSICAL.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn2

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="EDUCFORM.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn1

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
copyName="MILITARY.db"
copyTB()
endmethod
```

Object : stselectPage.#Box265.#Box266.departmentBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
copyName="STUDENT.db"
copyTB()
endmethod
```

Object : strestorePage

MethodName : arrive

Source :

```
method arrive(var eventInfo MoveEvent)
if Not isMaximized() then
        maximize()
endIf
setTitle("Student Restore Menu")
pageMenu()
endmethod
```

Object : strestorePage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="Restore &All tables"    :strestoretableBttn.pushbutton()
 case mc ="Select &Table to resotre":stselecttableBttn.pushbutton()
 case mc ="&Student disk menu"     :returnBttn9.pushbutton()
endswitch
endmethod
```

Object : strestorePage

MethodName : proc

Source :

```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("Restore &All tables")
dropMenu1.addText("Select &Table to resotre")
pageMenu.addPopUp("&Selection",dropmenu1)

dropMenu2.addText("&Student disk menu")
pageMenu.addPopUp("&Return",dropMenu2)
```

```
pageMenu.show()
endproc
```

Object : strestorePage.#Box255.#Box256.returnBttn9

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        stdiskPage.moveTo()
endmethod
```

Object : strestorePage.#Box255.#Box256.strestoretableBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        fs FileSystem
        tmpTb Table
        tableName       Array[] String
endVar
if msgQuestion("Please Confirm ! ...",
        "Are you sure that you want to restore all the "+
  "Tables form the STUDENT Diskette copy") = "Yes" then
        tableName.setSize(11)
        tableName[1]="ADDDROP.db"
        tableName[2]="CHILDREN.db"
        tableName[3]="EDUCFORM.db"
        tableName[4]="ENROLLED.db"
        tableName[5]="MILITARY.db"
        tableName[6]="PHYSICAL.db"
        tableName[7]="SPOUSE.db"
        tableName[8]="SPACTIVE.db"
        tableName[9]="ST-SCHED.db"
        tableName[10]="STUDENT.db"
        tableName[11]="THESIS.db"
        for i from 1 to 11
                tmpTb.attach(newDrive+tableName[i])
                tmpTb.add(tableName[i],True,True)
                message("Restore form <"+newDrive+tableName[i]+
                                "> To C:\\pdoxwin\\smdb\\"+tableName[i])
        endFor
endif
endmethod
```

Object : strestorePage.#Box255.#Box256.stselecttableBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
stselectPage.moveTo()
endmethod
```

Object : stdiskPage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&A:" :aBttn.pushbutton()
 case mc ="&B:" :bBttn.pushbutton()
 case mc ="&C:" :cBttn.pushbutton()
 case mc ="&Create new diskette"     :newdiskBttn.pushbutton()
 case mc ="&Quarterly Update"        :qrtlyBttn.pushbutton()
 case mc ="&Restore from student disk":strestoreBttn.pushbutton()
 case mc ="&Main menu"               :returnBttn8.pushbutton()
endswitch
endmethod
```

Object : stdiskPage

MethodName : proc

Source :
```
proc PageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar
dropMenu1.addText("&A:")
dropMenu1.addText("&B:")
dropMenu1.addText("&C:")
pageMenu.addPopUp("&Drive",dropmenu1)

dropMenu2.addText("&Create new diskette")
dropMenu2.addText("&Quarterly Update")
dropMenu2.addText("&Restore from student disk")
pageMenu.addPopUp("&Selection",dropmenu2)

dropMenu3.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu3)

pageMenu.show()


endproc
```

Object : stdiskPage.#Box5.#Group245.returnBttn8

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
MainPage.moveTo()
```

```
endmethod
```

Object : stdiskPage.#Box5.#Group245.strestoreBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if not newDrive.isBlank() then
        strestorePage.moveTo()
else
        msgStop("WARNING !","you must specify the Drive label")
endIf
endmethod
```

Object : stdiskPage.#Box5.#Group245.qrtlyBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        fs                      FileSystem
        stPath          String
        tableName       Array[] String
endVar
stpath="C:\\pdoxwin\\smdb\\stdisk\\"
if not newDrive.isBlank() then
        tableName.setSize(11)
        tableName[1]="SCHEDULE.db"
        tableName[2]="WEEKDAY.db"
        tableName[3]="PERIODS.db"
        tableName[4]="COURSES.db"
        tableName[5]="CURRTB.db"
        tableName[6]="CURRCOUR.db"
        tableName[7]="DEPARTMT.db"
        tableName[8]="SUBSPEC.db"
        tableName[9]="FACULTY.db"
        tableName[10]="PHYPOINT.db"
        tableName[11]="REQUIRED.db"
        for i from 1 to 11
                copy(tableName[i],newDrive+tableName[i])
                message("copy < "+tableName[i]  +
                                " > to < "+newDrive+tableName[i]+" >")
        endFor
        copy(stpath+"ST-SCHED.db",newDrive+"ST-SCHED.DB")
        message("copy < "+stPath+"ST-SCHED.db"  +
                        " > to < "+newDrive+"ST-SCHED.db")
        copy(stpath+"ADDDROP.db",newDrive+"ADDDROP")
        message("copy < "+stPath+"ADDDROP.db"  +
                        " > to < "+newDrive+"ADDDROP.db")
else
        msgStop("WARNING! Error","You must specify the Drive label")
endIf

endmethod
```

Object : stdiskPage.#Box5.#Group245.newdiskBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
 fs FileSystem
 mainPath String
endVar
if not newDrive.isBlank() then
        if msgQuestion("Please Confirm ! ...",
        "All the data in the destination disk will LOST!") = "Yes" then
                l=fs.makeDir("b:\\smdb")
                message("Status",iif(l,"New directory created","makeDir failure"))
                mainPath="C:\\pdoxwin\\smdb\\stdisk\\"
                if fs.findFirst("C:\\pdoxwin\\smdb\\stdisk\\*.fdl") then
                        fs.copy(mainPath+fs.name(),newDrive+fs.name())
                        message("copy < "+mainPath+fs.name()  +
                                                " > to < "+newDrive+fs.name())
                        while fs.findNext()
                                fs.copy(mainPath+fs.name(),newDrive+fs.name())
                                message("copy < "+mainPath+fs.name()  +
                                                        " > to < "+newDrive+fs.name())
                        endWhile
                endIf
                if fs.findFirst("C:\\pdoxwin\\smdb\\stdisk\\*.db") then
                        copy(mainPath+fs.name(),newDrive+fs.name())
                        message("copy < "+mainPath+fs.name()  +
                                                " > to < "+newDrive+fs.name())
                        while fs.findNext()
                                copy(mainPath+fs.name(),newDrive+fs.name())
                                message("copy < "+mainPath+fs.name()  +
                                                        " > to < "+newDrive+fs.name())
                        endWhile
                endIf
                fs.copy(mainPath+"smdb.ssl",newDrive+"smdb.ssl")
                fs.copy(mainPath+"smdb.bat",newDrive.subStr(1,3)+"smdb.bat")
                qrtlyBttn.pushButton()
        endif
else
        msgStop("WARNING! Error",
                                "you must specify the Drive label")
endIf
endmethod
```

Object : stdiskPage.#Box5.#Box7.#Button15

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
newDrive ="C:\\smdb"
endmethod
```

Object : stdiskPage.#Box5.#Box7.#Button10

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
newDrive ="B:\\smdb\\"
endmethod
```

Object : stdiskPage.#Box5.#Box7.#Button8

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
newDrive ="A:\\smdb"
endmethod
```

Object : selectPage

MethodName : copyTB

Source :
```
method copyTB()
Var
        tmpTb Table
endVar
if msgQuestion("Please Confirm ! ...",
                "Do you want to restore the < "+copyName+" > table form the back up
copy") = "Yes" then
        tmpTb.attach(newDrive+copyName)
        tmpTb.add(copyName,True,True)
endif
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn24

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="SUBSPEC.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn23

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="TAKENBY.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn22

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="THESIS.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn21

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="WARRINING.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn20

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="WEEKDAY.db"
copyTB()
endmethod

Object : selectPage.#Box181.#Box185.departmentBttn14

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="ST_SCHED.db"
copyTB()
endmethod

Object : selectPage.#Box181.#Box185.departmentBttn13

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="SPACTIVE.db"
copyTB()
endmethod

Object : selectPage.#Box181.#Box185.departmentBttn12

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="SPOUSE.db"
copyTB()
endmethod

Object : selectPage.#Box181.#Box185.departmentBttn11

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="SCHEDULE.db"
copyTB()
endmethod

Object : selectPage.#Box181.#Box185.departmentBttn10

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="REQUIRED.db"
copyTB()
endmethod

Object : selectPage.#Box181.#Box185.departmentBttn19

MethodName : pushButton

Source : method pushButton(var eventInfo Event)
copyName="ENROLLED.db"
copyTB()

```
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn18

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="FACULTY.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn17

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="PASSWD.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn16

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="PERIODS.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn15

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyname="PHYPOINT.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn9

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="ADDDROP.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn8

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyname="COURSES.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn7

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copynmae="CURRCOUR.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn6

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        tmpTb Table
endVar
if msgQuestion("Please Confirm ! ...",
                                        "Do you want to restore the Curriculum table form the back
up copy") = "Yes" then
        tmpTb.attach(newDrive+"CURRTB.db")
        tmpTb.add("CURRTB.db",True,True)
endif
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn5

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="DEPARTMT.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn4

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="CHLIDREN.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn3

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="PHYSICAL.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn2

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="EDUCFORM.db"
copyTB()
```

```
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn1

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="MILITARY.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box185.departmentBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
copyName="STUDENT.db"
copyTB()
endmethod
```

Object : selectPage.#Box181.#Box182.returnBttn8

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        restorePage.moveTo()
endmethod
```

Object : restorePage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="Restore &All tables"     :restoretableBttn.pushbutton()
 case mc ="Select &Table to resotre":selecttableBttn.pushbutton()
 case mc ="&Back up menu"           :returnBttn7.pushbutton()
endswitch
endmethod
```

Object : restorePage

MethodName : proc

Source :
```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("Restore &All tables")
```

```
dropMenu1.addText("Select &Table to resotre")
pageMenu.addPopUp("&Selection",dropmenu1)

dropMenu2.addText("&Back up menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()

endproc
```

Object : restorePage.#Box170.#Box174.restoretableBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        fs FileSystem
        tmpTb Table
endVar
if msgQuestion("Please Confirm ! ...",
        "Are you sure that you want to restore all the "+
  "Tables form the back up copy") = "Yes" then
        if fs.findFirst(newDrive+"*.db") then
                tmpTb.attach(newDrive+fs.name())
                tmpTb.add(fs.name(),True,True)
                message("Restore form <"+newDrive+fs.name()+
                                        "> To C:\\"+fs.name())
                while fs.findNext()
                        tmpTb.attach(newDrive+fs.name())
                        tmpTb.add(fs.name(),True,True)
                        message("Restore form <"+newDrive+fs.name()+
                                        "> To C:\\"+fs.name())

                endWhile
        endIf
endif
endmethod
```

Object : restorePage.#Box170.#Box174.selecttableBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
selectPage.moveTo()
endmethod
```

Object : restorePage.#Box170.#Box171.returnBttn7

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        backupPage.moveTo()
endmethod
```

Object : backupPage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&A:" :aBttn.pushbutton()
 case mc ="&B:" :bBttn.pushbutton()
 case mc ="&C:" :cBttn.pushbutton()
 case mc ="&Daily back up"       :dailyBttn.pushbutton()
 case mc ="&Graduates back up"   :gradbackBttn.pushbutton()
 case mc ="&Restore from back ups":retoreBttn.pushbutton()
 case mc ="&Main menu"           :returnBttn6.pushbutton()
endswitch
endmethod
```

Object : backupPage

MethodName : proc

Source :

```
proc PageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2, dropMenu3 popUpMenu
endVar
dropMenu1.addText("&A:")
dropMenu1.addText("&B:")
dropMenu1.addText("&C:")
pageMenu.addPopUp("&Drive",dropmenu1)

dropMenu2.addText("&Daily back up")
dropMenu2.addText("&Graduates back up")
dropMenu2.addText("&Restore from back ups")
pageMenu.addPopUp("&Selection",dropmenu2)

dropMenu3.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu3)

pageMenu.show()


endproc
```

Object : backupPage.bigbox.returnBttn6

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
MainPage.moveTo()
endmethod
```

Object : backupPage.bigbox.returnBttn6.#Text168

MethodName : action

Source :
```
method action(var eventInfo ActionEvent)
mainPage.moveTo()
endmethod
```

Object : backupPage.bigbox.#Group156.retoreBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if not newDrive.isBlank() then
        restorePage.moveTo()
else
        msgStop("WARNING !","you must specify the Drive label")
endIf
endmethod
```

Object : backupPage.bigbox.#Group156.gradbackBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        tmpTb Table
endVar
if not newDrive.isBlank() then
;
else
        msgStop("WARNING! Error","you must specify the Drive label")
endIf
endmethod
```

Object : backupPage.bigbox.#Group156.dailyBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        fs FileSystem
endVar
if not newDrive.isBlank() then
        if msgQuestion("Please Confirm ! ...",
                "Are you sure that you want to back up all files") = "Yes" then
                if fs.findFirst("*.db") then
                        copy(fs.name(),newDrive+fs.name())
                        while fs.findNext()
                                copy(fs.name(),newDrive+fs.name())
                        endWhile
                endIf
        endif
else
        bigbox.visible="False"
        msgStop("WARNING! Error",
                        "you must specify the Drive label")
        bigbox.visible="True"
endIf
endmethod
```

Object : backupPage.bigbox.#Box149.cBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
newDrive ="C:\\"
endmethod
```

Object : backupPage.bigbox.#Box149.bBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
newDrive ="B:\\"
endmethod
```

Object : backupPage.bigbox.#Box149.#Button150

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
newDrive ="A:\\"
endmethod
```

Object : codePage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&Department form" :departmentBttn.pushbutton()
 case mc ="&Subspeciality form":subspecialBttn.pushbutton()
 case mc ="&Physical Point Table":phypointBttn.pushbutton()
 case mc ="&Main menu":returnBttn5.pushbutton()
endswitch
endmethod
```

Object : codePage

MethodName : proc

Source :
```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("&Department form")
dropMenu1.addText("&Subspeciality form")
dropMenu1.addText("&Physical Point Table")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()

endproc
```

```
Object :        codePage.#Box133.#Box140.phypointBttn
MethodName :    pushButton
Source :        method pushButton(var eventInfo Event)
                Var
                        openForm Form
                endVar
                        openForm.open("PHYPOINT.fdl")
                        disableDefault
                endmethod

Object :        codePage.#Box133.#Box140.departmentBttn
MethodName :    pushButton
Source :        method pushButton(var eventInfo Event)
                Var
                        openForm Form
                endVar
                        openForm.open("DEPART.fdl")
                        disableDefault
                endmethod

Object :        codePage.#Box133.#Box140.subspecialBttn
MethodName :    pushButton
Source :        method pushButton(var eventInfo Event)
                Var
                        openForm Form
                endVar
                        openForm.open("SUBSPECI.fdl")
                        disableDefault
                endmethod

Object :        codePage.#Box133.#Box134.returnBttn5
MethodName :    pushButton
Source :        method pushButton(var eventInfo Event)
                        MainPage.moveTo()
                endmethod

Object :        performancePage
MethodName :    menuAction
Source :        method menuAction(var eventInfo MenuEvent)
                Var
                        mc String
                endVar

                mc = eventInfo.menuChoice()
                switch
                 case mc ="&Individual Letter" :individualBttn.pushbutton()
                 case mc ="&Group     Letters":groupBttn.pushbutton()
                 case mc ="&Reports menu":returnBttn4.pushbutton()
                endswitch
                endmethod
```

Object : performancePage

MethodName : proc

Source :
```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("&Individual Letter")
dropMenu1.addText("&Group     Letters")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Reports menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()

endproc
```

Object : performancePage.#Box125.#Box139.individualBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("WARNSING.fdl")
        disableDefault
endmethod
```

Object : performancePage.#Box125.#Box139.groupBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("WARRNING.fdl")
        disableDefault
endmethod
```

Object : performancePage.#Box125.#Box129.returnBttn4

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        reportsPage.moveTo()
endmethod
```

Object : reportspage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
```

```
            mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&Performance Letters"  :performanceBttn.pushbutton()
 case mc ="&Graduates List"       :graduateBttn.pushbutton()
 case mc ="&Diploma Mailing Label":labelBttn.pushbutton()
 case mc ="&Academic Reports"     :acadreportBttn.pushbutton()
 case mc ="&Main menu"            :returnBttn3.pushbutton()
endswitch
endmethod
```

Object : reportspage

MethodName : proc

Source :

```
proc PageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("&Performance Letters")
dropMenu1.addText("&Graduates List")
dropMenu1.addText("&Diploma Mailing Label")
dropMenu1.addText("&Academic Reports")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()


endproc
```

Object : reportspage.#Box18.#Box114.labelBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Report
endVar
        openForm.open("LABEL.rdl")
        disableDefault
endmethod
```

Object : reportspage.#Box18.#Box114.graduateBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Report
endVar
        openForm.open("GRADUATE.rdl")
        disableDefault
```

```
endmethod
```

Object : reportspage.#Box18.#Box114.performanceBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
performancePage.moveTo()
endmethod
```

Object : reportspage.#Box18.#Box111.returnBttn3

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        MainPage.moveTo()
endmethod
```

Object : mstSchedulePage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&Master Schedule form" :masterschedBttn.pushbutton()
 case mc ="&Course Schedule form" :courseschedBttn.pushbutton()
 case mc ="&Main menu":returnBttn3.pushbutton()
endswitch
endmethod
```

Object : mstSchedulePage

MethodName : proc

Source :
```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("&Master Schedule form")
dropMenu1.addText("&Course Schedule form")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()
```

```
endproc
```

Object : mstSchedulePage.#Box62.#Box99.masterschedBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("MSTSCHED.fdl")
        disableDefault
endmethod
```

Object : mstSchedulePage.#Box62.#Box99.courseschedBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("TAUGHTBY.fdl")
        disableDefault
endmethod
```

Object : mstSchedulePage.#Box62.#Box96.returnBttn3

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        MainPage.moveTo()
endmethod
```

Object : stSchedulePage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
```

```
 case mc ="&Curriculum Enrolled form" :currenrollBttn.pushbutton()
 case mc ="&Student Schedule form":stscheduleBttn.pushbutton()
 case mc ="&Add/Drop form":addropBttn.pushbutton()
 case mc ="Academic &Records form":academicBttn.pushbutton()
 case mc ="&Main menu":returnBttn2.pushbutton()
endswitch
endmethod
```

Object : stSchedulePage

MethodName : proc

Source :

```
proc PageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("&Curriculum Enrolled form")
dropMenu1.addText("&Student Schedule form")
dropMenu1.addText("&Add/Drop form")
dropMenu1.addText("Academic &Records form")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()

endproc
```

Object : stSchedulePage.#Box110.#Box95.academicBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("TAKENBY.fdl")
        disableDefault
endmethod
```

Object : stSchedulePage.#Box110.#Box95.addropBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("ADDDROP.fdl")
        disableDefault
endmethod
```

Object : stSchedulePage.#Box110.#Box95.stscheduleBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("STSCHED.fdl")
        disableDefault
endmethod
```

Object : stSchedulePage.#Box110.#Box95.currenrollBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("enrolled.fdl")
        disableDefault
endmethod
```

Object : stSchedulePage.#Box110.#Box92.returnBttn2

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        MainPage.moveTo()
endmethod
```

Object : curriculumPage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&New Curriculum form" :curriculumBttn.pushbutton()
 case mc ="&Curriculum courses form":currcoursesBttn.pushbutton()
 case mc ="&Main menu":returnBttn1.pushbutton()
endswitch
endmethod
```

Object : curriculumPage

MethodName : proc

Source :
```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar
```

```
dropMenu1.addText("&New Curriculum form")
dropMenu1.addText("&Curriculum courses form")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()


endproc
```

Object : curriculumPage.#Box45.#Box54.curriculumBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("CURRICUL.fdl")
        disableDefault
endmethod
```

Object : curriculumPage.#Box45.#Box54.currcoursesBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("CURRCOUR.fdl")
        disableDefault
endmethod
```

Object : curriculumPage.#Box45.#Box51.returnBttn1

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        MainPage.moveTo()
endmethod
```

Object : studentpage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
```

```
switch
 case mc ="&Personnel form" :personnelBttn.pushbutton()
 case mc ="&Military form":militaryBttn.pushbutton()
 case mc ="&Education form":educationBttn.pushbutton()
 case mc ="P&hysical form":physicalBttn.pushbutton()
 case mc ="&Spouse from":spouseBttn.pushbutton()
 case mc ="&Children form":childrenBttn.pushbutton()
 case mc ="Change Pass&word":passwdBttn.pushbutton()
 case mc ="&Main menu":returnBttn.pushbutton()
endswitch
endmethod
```

Object : studentpage

MethodName : proc

Source :

```
proc PageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu1.addText("&Personnel form")
dropMenu1.addText("&Military form")
dropMenu1.addText("&Education form")
dropMenu1.addText("P&hysical form")
dropMenu1.addText("&Spouse from")
dropMenu1.addText("&Children form")
dropMenu1.addSeparator()
dropMenu1.addText("Change Pass&word")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Main menu")
pageMenu.addPopUp("&Return",dropMenu2)

pageMenu.show()

endproc
```

Object : studentpage.#Box87.#Box72.personnelBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("personel.fdl")
        disableDefault
endmethod
```

Object : studentpage.#Box87.#Box72.militaryBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("MILITARY.fdl")
        disableDefault
endmethod
```

Object : studentpage.#Box87.#Box72.spouseBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("SPOUSE.fdl")
        disableDefault
endmethod
```

Object : studentpage.#Box87.#Box72.physicalBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("PHYSICAL.fdl")
        disableDefault
endmethod
```

Object : studentpage.#Box87.#Box72.childrenBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("CHILDERN.fdl")
        disableDefault
endmethod
```

Object : studentpage.#Box87.#Box72.educationBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("EDUCFORM.fdl")
        disableDefault
endmethod
```

Object : studentpage.#Box87.#Box69.returnBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
        MainPage.moveTo()
endmethod
```

Object : studentpage.#Box87.#Box42.passwdBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
        openForm.open("PASSCHG.fdl")
        disableDefault
endmethod
```

Object : MainPage

MethodName : menuAction

Source :
```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar

mc = eventInfo.menuChoice()
switch
 case mc ="&Student" :StudentBttn.pushbutton()
 case mc ="Student Sche&dule menu":StScheduleBttn.pushbutton()
 case mc ="&Main Schedule menu":ScheduleBttn.pushbutton()
 case mc ="&Courses":CoursesBttn.pushbutton()
 case mc ="&Faculty":FacultyBttn.pushbutton()
 case mc ="Cu&rriculum menu":CurriculumBttn.pushbutton()
 case mc ="&Thesis":ThesisBttn.pushbutton()
 case mc ="&Codes menu":codesBttn.pushbutton()
 case mc ="&Add new user":useridBttn.pushbutton()
 case mc ="&End of quater":endquarterBttn.pushbutton()
 case mc ="&Genrate Reports":reportsBttn.pushbutton()
 case mc ="&Create student Diskette":stdiskBttn.pushbutton()
 case mc ="&Back Ups":backupBttn.pushbutton()
 case mc ="&Quit":ExitBttn.pushbutton()
endswitch
endmethod
```

Object : MainPage

MethodName : proc

Source :
```
proc PageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2,dropMenu3 popUpMenu
endVar

dropMenu1.addText("&Student")
dropMenu1.addText("Student Sche&dule menu")
dropMenu1.addText("&Main Schedule menu")
dropMenu1.addText("&Courses")
```

```
dropMenu1.addText("&Faculty")
dropMenu1.addText("Cu&rriculum menu")
dropMenu1.addText("&Thesis")
dropMenu1.addText("&Codes menu")
pageMenu.addPopUp("Se&lection",dropmenu1)

dropMenu2.addText("&Add new user")
dropMenu2.addText("&End of quarter")
dropMenu2.addText("&Genrate Reports")
dropMenu2.addText("&Create student Diskette")
dropMenu2.addText("&Back Ups")
pageMenu.addPopUp("&Procedures",dropmenu2)

dropMenu3.addText("&Quit")
pageMenu.addPopUp("&Quit",dropMenu3)

pageMenu.show()

endproc
```

Object : MainPage.#Box25.ExitBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        PassWdField=""
        UserIdField = ""
        UserIdField.moveTo()
endmethod
```

Object : MainPage.#Box243.CurriculumBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        curriculumPage.moveTo()
endif
endmethod
```

Object : MainPage.#Box243.StScheduleBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        stSchedulePage.moveTo()
```

```
endif
endmethod
```

Object : MainPage.#Box243.ScheduleBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        mstSchedulePage.moveTo()
endif
endmethod
```

Object : MainPage.#Box243.CoursesBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        disableDefault
        hide()
        openForm.open("COURSES.fdl")
endif
endmethod
```

Object : MainPage.#Box243.FacultyBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        openForm.open("FACULTY.fdl")
        disableDefault
endif
endmethod
```

Object : MainPage.#Box243.StudentBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
        studentPage.moveTo()
endmethod
```

Object : MainPage.#Box243.CodesBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        codePage.moveTo()
endif
endmethod
```

Object : MainPage.#Box243.ThesisBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        disableDefault
        hide()
        openForm.open("THESIS.fdl")
endif
endmethod
```

Object : MainPage.#Box41.#Box44.#Group106.useridBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
Var
        openForm Form
endVar
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        openForm.open("PASSWD.fdl")
        disableDefault
endif
endmethod
```

Object : MainPage.#Box41.#Box44.#Group106.stdiskBttn

MethodName : pushButton

Source :
```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        stdiskPage.moveTo()
endif
endmethod
```

Object : MainPage.#Box41.#Box44.backupBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        backupPage.moveTo()
endif
endmethod
```

Object : MainPage.#Box41.#Box44.endquarterBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
Var
        studentTC, stschedTC        TCursor
        acadTC, courseTC, addTC     TCursor
endVar
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
if msgQuestion("WARRNING!","This routine will update the Academic record of"+
    " all the student and Clear the Student Schedule, Add/drop tables")="Yes"  then
        addTC.open("ADDDROP.db")
        if studentTC.open("STUDENT.db")  and
                stschedTC.open("ST-SCHED.db") and
                acadTC.open("TAKENBY.db")              and
                courseTC.open("COURSES.db")  then
                studentTC.home()
                for i from 1 to studentTC.nRecords()
                        stschedTC.setFilter(studentTC.SSN)
                        if NOT stschedTC.isEmpty() then
                                acadTC.edit()
                                for j from 1 to stschedTC.nRecords()
                                        acadTC.insertRecord()
                                        acadTC.SSN=stschedTC.SSN
                                        acadTC."QUARTER
ORDER"=stschedTC."QUARTER ORDER"
                                        acadTC."COURSE NO"=stschedTC."COURSE
NO"
                                        acadTC."COURSE TYPE"=stschedTC."S_TYPE"
                                        if courseTC.locate("COURSE
NO",stschedTC."COURSE NO") then
                                                acadTC."Credit"=courseTC.CREDIT
                                                acadTC."Lab"=courseTC.LAB
                                        endIf
                                        stschedTC.nextRecord()
                                endFor
                                acadTC.endEdit()
                        endIf
                        stschedTC.setFilter()
                        studentTC.nextRecord()
                endFor
                stschedTC.empty()
                addTC.empty()
                studentTC.close()
                stschedTC.close()
                acadTC.close()
                courseTC.close()
                addTC.close()
        endIf
```

```
            noteG1=195
            noteG#1=207
            noteA2=220
            noteA#2=234
            noteB2=249
            noteC2=265
            noteC#2=282
            noteD2=300
endConst
sound(noteA1,200)
sound(noteD1,150)
sound(noteF#1,50)
sound(noteA2,100)
sound(noteB2,100)
sound(noteA2,150)
endmethod
```

Object : PassPage.#Box105.PassWdField

MethodName : action

Source :

```
method action(var eventInfo ActionEvent)
Var
        tc TCursor
        passWord String
endVar
tc.Open("PASSWD.db")
if not tc.Locate("UserId",UserIdField.upper()) then
        MsgStop("PASSWORD","User Id <"+UserIdField.upper()+"> NOT found on file")
        UserIdField = ""
        UserIdField.moveTo()
        disableDefault
else
        passWord = tc.PassWd
        if passWord <> PassWdField then
                soundPY()
                MsgStop("PASSWORD","Wrong Password")
                PassWdField=""
                UserIdField = ""
          UserIdField.moveTo()
                disableDefault
        else
                userCategory = tc."Category".value
                MainPage.moveTo()
                disableDefault
```

```
endmethod
```

Object : PassPage

MethodName : menuAction

Source :

```
method menuAction(var eventInfo MenuEvent)
Var
        mc String
endVar
mc = eventInfo.menuChoice()
switch
        case mc ="&Exit to System":self.close()
endswitch                                                    ;exit()
endmethod
```

Object : PassPage

MethodName : proc

Source :

```
proc pageMenu()
Var
        pageMenu Menu
        dropMenu1, dropMenu2 popUpMenu
endVar

dropMenu2.addText("&Exit to System")
pageMenu.addPopUp("&Exit",dropMenu2)

pageMenu.show()

endproc
```

Object : PassPage

MethodName : soundPY

Source :

```
method soundPY()
Var
qnote,octave, note longint
power Number
endVar

const
        noteA1 = 110
        noteA#1=116
        noteB1=123
        noteC1=130
        noteC#1=138
        noteD1=146
        noteD#1=155
        noteE1=164
        noteF1=174
        noteF#1=184
```

```
endIf
endIf
endmethod
```

Object : MainPage.#Box41.#Box44.ReportsBttn

MethodName : pushButton

Source :

```
method pushButton(var eventInfo Event)
if userCategory = "STUDENT" then
        msgInfo("CAUTION","You are NOT allowed to access <"+self.LabelText+"> form ")
else
        reportsPage.moveTo()
endif
```

```
                endIf
            endIf
;endWhile
tc.close()
endmethod
```

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, Virginia 22304-6145

2. Library, Code 052 — 2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. Systems Management, Code 36 — 5
   Naval Postgraduate School
   Monterey, California 93943-5002

4. LCDR William B. Short, Code SM/SH — 1
   Department of Systems Management
   Naval Postgraduate School
   Monterey, California 93943-5002

5. Shu Liao, Code SM/LC — 1
   Department of Systems Management
   Naval Postgraduate School
   Monterey, California 93943-5002

6. Barry Hubbard — 1
   4904 Holly Crest way
   Fair Oaks, California 95628

7. Sufian I Althawadi — 1
   P.O. Box 1969
   Manama-Bahrain